

CCCCCCCCCCCC	0000000000	BBBBBBBBBBBB	RRRRRRRRRR	TTTTTTTTTTTT	LLL
CCCCCCCCCCCC	0000000000	BBBBBBBBBBBB	RRRRRRRRRR	TTTTTTTTTTTT	LLL
CCCCCCCCCCCC	0000000000	BBBBBBBBBBBB	RRRRRRRRRR	TTTTTTTTTTTT	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCC	000	000 BBB	BBB RRR	RRR	LLL
CCCCCCCCCCCC	0000000000	BBBBBBBBBBBB	RRR	RRR	TTT
CCCCCCCCCCCC	0000000000	BBBBBBBBBBBB	RRR	RRR	TTT
CCCCCCCCCCCC	0000000000	BBBBBBBBBBBB	RRR	RRR	TTT

CCCCCCCC	000000	BBBBBBBB	EEEEEEEE	SSSSSS	CCCCCCCC	GGGGGGGG	EEEEEEEE	NN	NN
CCCCCCCC	000000	BBBBBBBB	EEEEEEEE	SSSSSS	CCCCCCCC	GGGGGGGG	EEEEEEEE	NN	NN
CC	00	00	BB	SS	CC	GG	EE	NN	NN
CC	00	00	BB	SS	CC	GG	EE	NN	NN
CC	00	00	BB	EE	CC	GG	EE	NN	NN
CC	00	00	BB	EE	SS	GG	EE	NNNN	NNNN
CC	00	00	BB	EE	SS	GG	EE	NNNN	NNNN
CC	00	00	BB	EE	SS	GG	EE	NN	NN
CC	00	00	BB	EE	SS	GG	EE	NN	NN
CC	00	00	BB	EE	SS	GG	EE	NN	NN
CC	00	00	BB	EE	SS	GG	EE	NN	NN
CC	00	00	BB	EE	SS	GG	EE	NN	NN
CC	00	00	BB	EE	SS	GG	EE	NN	NN
CC	00	00	BB	EE	SS	GG	EE	NN	NN
CC	00	00	BB	EE	SS	GG	EE	NN	NN
CCCCCCCC	000000	BBBBBBBB	EEEEEEEE	SSSSSS	CCCCCCCC	GGGGGG	EEEEEEEE	NN	NN
CCCCCCCC	000000	BBBBBBBB	EEEEEEEE	SSSSSS	CCCCCCCC	GGGGGG	EEEEEEEE	NN	NN

LL		SSSSSSSS
LL		SSSSSSSS
LL		SS
LLLLLLLL		SSSSSSSS
LLLLLLLL		SSSSSSSS

```
1 0001 0 XTITLE 'COB$SESCAPE GENERATOR - Escape sequence generator for screen mgmt'  
2 0002 0 MODULE COB$SESCAPE GENERATOR (  
3 0003 0 IDENT = '1-003' ! File: COBESCGEN.B32 Edit: STAN1003  
4 0004 0 ) =  
5 0005 1 BEGIN  
6 0006 1 *****  
7 0007 1 *  
8 0008 1 *  
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
11 0011 1 * ALL RIGHTS RESERVED.  
12 0012 1 *  
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
18 0018 1 * TRANSFERRED.  
19 0019 1 *  
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
22 0022 1 * CORPORATION.  
23 0023 1 *  
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
26 0026 1 *  
27 0027 1 *  
28 0028 1 *****  
29 0029 1 .  
30 0030 1  
31 0031 1 ++  
32 0032 1 FACILITY: General Utility Library  
33 0033 1  
34 0034 1 ABSTRACT:  
35 0035 1  
36 0036 1 This module contains routines which return a device-specific  
37 0037 1 escape sequence to perform a specified function.  
38 0038 1  
39 0039 1 These are low level routines: the burden of validity checking  
40 0040 1 is on the caller. For example, buffers are allocated by the caller,  
41 0041 1 and these routines do not check for overflowing the buffers bounds.  
42 0042 1 If the device is not a video terminal, no escape sequence will be  
43 0043 1 generated, and the routine will return with a success status.  
44 0044 1  
45 0045 1 ENVIRONMENT: User mode, Shared Library routines.  
46 0046 1  
47 0047 1 AUTHOR: P. Levesque, CREATION DATE: 7-Mar-1983  
48 0048 1  
49 0049 1 MODIFIED BY:  
50 0050 1  
51 0051 1 1-001 - Original. PLL 7-Mar-1983  
52 0052 1 1-002 - Add COB$SET ATTRIBUTES ONLY.  
53 0053 1 Fix call to COB$SET_CURSOR_ABS_R4 in COB$SET_CURSOR_REL.  
54 0054 1 Fix to COB$SET_CURSOR_REL. If we are at the 1st column and the  
55 0055 1 previous character was a <CR>, then the terminal driver may give  
56 0056 1 us a 'free' <LF> on our next operation if it is a read. To avoid  
57 0057 1 the problem, just make sure <CR> is not the last thing in the
```

COB\$ESCAPE\_GEN COB\$ESCAPE\_GENERATOR - Escape sequence generat <sup>K 10</sup>  
1-003 16-Sep-1984 00:06:34 14-Sep-1984 12:10:44 VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBESCGEN.B32;1

Page 2  
(1)

58 0058 1 | output buffer.  
59 0059 1 | Rename module from SMG\$ESCAPE\_GENERATOR to COB\$ESCAPE GENERATOR.  
60 0060 1 | LGB 20-FEB-1984  
61 0061 1 | 1-003 - Removed informational errors. STAN 24-Jul-1984.  
62 0062 1 |  
63 0063 1 | --

```
65 0064 1 XSBTTL 'Declarations'  
66 0065 1  
67 0066 1 SWITCHES:  
68 0067 1  
69 0068 1  
70 0069 1  
71 0070 1 LINKAGES:  
72 0071 1  
73 0072 1 NONE  
74 0073 1  
75 0074 1 INCLUDE FILES:  
76 0075 1  
77 0076 1 REQUIRE 'RTLIN:COBPROLOG'; ! Defines psects, macros, &  
78 1593 1 terminal defs  
79 1594 1 REQUIRE 'RTLIN:COBLNK'; ! Linkages  
80 1669 1  
81 1670 1 TABLE OF CONTENTS:  
82 1671 1  
83 1672 1  
84 1673 1 FORWARD ROUTINE  
85 1674 1  
86 1675 1 COB$DOWN_SCROLL_R2 : COB$ESC_R2_LNK; ! Create downscroll sequence  
87 1676 1 COB$ERASE_LINE_R2 : COB$ESC_R2_LNK; ! Create erase line sequence  
88 1677 1 COB$ERASE_PAGE_R2 : COB$ESC_R2_LNK; ! Create erase page sequence  
89 1678 1 COB$ERASE_WHOLE_LINE_R2 : COB$ESC_R2_LNK; ! Create erase whole line sequence  
90 1679 1 COB$ERASE_WHOLE_PAGE_R2 : COB$ESC_R2_LNK; ! Create erase whole page sequence  
91 1680 1 COB$SET_ATTRIBUTES; ! Create set attributes sequences w text  
92 1681 1 COB$SET_ATTRIBUTES_ONLY; ! Create set attributes sequences w no text  
93 1682 1 COB$SET_CURSOR_ABS_R4 : COB$ESC_R4_LNK; ! Create absolute set cursor sequence  
94 1683 1 COB$SET_CURSOR_REL; ! Create relative set cursor sequence  
95 1684 1 COB$SETUP_TERM_TYPE; ! Setup terminal type for COB$ calls  
96 1685 1 COB$UP_SCROLL_R2 : COB$ESC_R2_LNK; ! Create upscroll sequence  
97 1686 1  
98 1687 1  
99 1688 1 MACROS:  
100 1689 1  
101 1690 1  
102 1691 1  
103 1692 1 EQUATED SYMBOLS:  
104 1693 1  
105 1694 1  
106 1695 1  
107 1696 1 FIELDS:  
108 1697 1  
109 1698 1 NONE  
110 1699 1  
111 1700 1 PSECTS:  
112 1701 1  
113 1702 1  
114 1703 1 OWN STORAGE:  
115 1704 1  
116 1705 1 NONE  
117 1706 1  
118 1707 1  
119 1708 1 EXTERNAL REFERENCES:  
120 1709 1  
121 1710 1
```

COB\$ESCAPE\_GEN COB\$ESCAPE\_GENERATOR - Escape sequence generat M 10  
1-003 Declarations 16-Sep-1984 00:06:34 14-Sep-1984 12:10:44 VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBESCGEN.B32;1 Page 4 (2)

```
122 1711 1 EXTERNAL ROUTINE
123 1712 1
124 1713 1 LIB$FREE_EF,
125 1714 1 LIB$GET_EF;
126 1715 1 ! free event flag number
127 1716 1 ! get event flag number
1716 1 !<BLF/PAGE>
```

```
129      1717 1 %SBTTL 'COB$DOWN_SCROLL_R2 - Create downscroll sequence'  
130      1718 1 GLOBAL ROUTINE COB$DOWN_SCROLL_R2 ( 16-Sep-1984 00:06:34  
131      1719 1 TERM_TYPE,  
132      1720 1 BUFFER,  
133      1721 1 CUR_SIZE  
134      1722 1 ) : COB$SESC_R2_LNK =  
135      1723 1 ++  
136      1724 1 FUNCTIONAL DESCRIPTION:  
137      1725 1  
138      1726 1 This routine generates the escape sequence for down scroll  
139      1727 1 and appends the string to a given output buffer.  
140      1728 1  
141      1729 1 CALLING SEQUENCE:  
142      1730 1  
143      1731 1 ret_status.wlc.v = COB$DOWN_SCROLL_R2 (TERM_TYPE.rl.v, BUFFER.mt.r,  
144      1732 1 CUR_SIZE.ml.r)  
145      1733 1  
146      1734 1 FORMAL PARAMETERS:  
147      1735 1  
148      1736 1 TERM_TYPE.rl.v terminal type  
149      1737 1 BUFFER.mt.r addr of buffer  
150      1738 1 CUR_SIZE.ml.r # bytes currently in buffer  
151      1739 1  
152      1740 1 IMPLICIT INPUTS:  
153      1741 1  
154      1742 1 NONE  
155      1743 1  
156      1744 1 IMPLICIT OUTPUTS:  
157      1745 1  
158      1746 1 NONE  
159      1747 1  
160      1748 1 COMPLETION STATUS:  
161      1749 1  
162      1750 1  
163      1751 1 SIDE EFFECTS:  
164      1752 1  
165      1753 1 NONE  
166      1754 1 ---  
167      1755 1  
168      1756 2 BEGIN  
169      1757 2  
170      1758 2 LOCAL  
171      1759 2 FREE_ADDR;  
172      1760 2  
173      1761 2 BIND  
174      1762 2 VT05_DOWN = UPLIT (BYTE (CR, VT05_CUP, NULL)),  
175      1763 2 VT52_DOWN = UPLIT (BYTE (ESC, VT52_DWN))  
176      1764 2 VT100_DOWN = UPLIT (BYTE (ESC, VT100_DWN));  
177      1765 2  
178      1766 2 FREE_ADDR = .BUFFER + ..CUR_SIZE;  
179      1767 2  
180      1768 2 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF  
181      1769 2 SET  
182      1770 2 [VT05]:  
183      1771 2 BEGIN  
184      1772 2 CH$MOVE (3, VT05_DOWN, .FREE_ADDR);  
185      1773 3 .CUR_SIZE = ..CUR_SIZE + 3;
```

```

    186      1774  2      END;
    187      1775  2
    188      1776  2
    189      1777  3      [VT52]:
    190      1778  3      BEGIN
    191      1779  3      CHSMOVE (2, VT52_DOWN, .FREE_ADDR);
    192      1780  2      .CUR_SIZE = ..CUR_SIZE + 2;
    193      1781  2      END;
    194      1782  2
    195      1783  3      [VT100]:
    196      1784  3      BEGIN
    197      1785  3      CHSMOVE (2, VT100_DOWN, .FREE_ADDR);
    198      1786  2      .CUR_SIZE = ..CUR_SIZE + 2;
    199      1787  2      END;
    200      1788  2      [HARDCOPY, UNKNOWN, VTFOREIGN]:
    201      1789  2      ;
    202      1790  2
    203      1791  2      [INRANGE, OUTRANGE]:
    204      1792  2      RETURN 0;           ! should never get here
    205      1793  2
    206      1794  2      TES;
    207      1795  2
    208      1796  2      RETURN (SSS_NORMAL);
    209      1797  2
    210      1798  1      END;

```

.TITLE COB\$ESCAPE\_GENERATOR COB\$ESCAPE\_GENERATOR - E  
 .IDENT 1-003\ escape sequence generat

.PSECT \_COB\$CODE,NOWRT, SHR, PIC.2

00	1A	0D	00000	P.AAA:	.BYTE	13, 26, 0
			00003		.BLKB	1
49	1B	00004	P.AAB:	.BYTE	27, 73	
		00006		.BLKB	2	
4D	1B	00008	P.AAC:	.BYTE	27, 77	

VT05_DOWN=	P.AAA
VT52_DOWN=	P.AAB
VT100_DOWN=	P.AAC
.EXTRN	LIB\$FREE_EF, LIB\$GET_EF

001F	0019	05	51	62	CO 00000	COB\$DOWN_SCROLL_R2::	1766 1768	
			00	50	CF 00003	ADDL2		TCUR_SIZE), FREE_ADDR
			000E	0026	00007 18:	CASEL		TERM-TYPE, #0, #5
			0026	0026	0000F	.WORD		6S-1S,- 2S-1S,- 3S-1S,- 4S-1S,- 6S-1S,- 6S-1S
61	18	00	DE	1C AF 00013	BRB	7S	1792 1772 1773 1768	
			62	F0 00015 28:	INSV	VT05_DOWN, #0, #24, (FREE_ADDR)		
			03	CO 0001B	ADDL2	#3, TCUR_SIZE		
			0D	11 0001E	BRB	6S		

COB\$ESCAPE\_GEN COB\$ESCAPE GENERATOR - Escape sequence generat 16-Sep-1984 00:06:34  
1-003 COB\$DOWN\_SCROLL\_R2 - Create downscroll sequenc 14-Sep-1984 12:10:44 C 11 VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBESCGEN.B32;1

Page 7  
(3)

61	D7	AF	B0	00020	3\$:	MOVW	VT52_DOWN, (FREE_ADDR)	:	1778
		04	11	00024		BRB	5\$		1779
61	D5	AF	B0	00026	4\$:	MOVW	VT100_DOWN, (FREE_ADDR)	:	1784
62		02	C0	0002A	5\$:	ADDL2	#2, (CUR_SIZE)		1785
50		01	D0	0002D	6\$:	MOVL	#1, R0		1796
			05	00030		RSB			
			50	D4	00031	7\$:	CLRL	R0	
					05	00033	RSB		1798

: Routine Size: 52 bytes. Routine Base: \_COB\$CODE + 000A

: 211 1799 1 !<BLF/PAGE>

```
1800 1 %SBTTL 'COB$SERASE LINE_R2 - Create erase line sequence'  
1801 1 GLOBAL ROUTINE COB$SERASE LINE_R2 (   
1802 1 TERM_TYPE,  
1803 1 BUFFER,  
1804 1 CUR_SIZE,  
1805 1 ) : COB$SESC_R2_LNK =  
1806 1  
1807 1 ++  
1808 1 FUNCTIONAL DESCRIPTION:  
1809 1  
1810 1 This routine generates the escape sequence for erasing a  
1811 1 line from the current cursor position. The string is  
1812 1 appended to the given output buffer.  
1813 1  
1814 1 CALLING SEQUENCE:  
1815 1  
1816 1 ret_status.wlc.v = COB$SERASE_LINE_R2 (TERM_TYPE.rl.v,  
1817 1 BUFFER.mt.r, CUR_SIZE.ml.r)  
1818 1  
1819 1 FORMAL PARAMETERS:  
1820 1  
1821 1 TERM_TYPE.rl.v terminal type  
1822 1 BUFFER.mt.r addr of buffer  
1823 1 CUR_SIZE.ml.r # bytes currently in buffer  
1824 1 updated to reflect erase seq added  
1825 1  
1826 1 IMPLICIT INPUTS:  
1827 1  
1828 1  
1829 1  
1830 1  
1831 1  
1832 1  
1833 1  
1834 1  
1835 1  
1836 1  
1837 1  
1838 1  
1839 1  
1840 1  
1841 2  
1842 2  
1843 2  
1844 2  
1845 2  
1846 2  
1847 2  
1848 2  
1849 2  
1850 2  
1851 2  
1852 2  
1853 2  
1854 2  
1855 2  
1856 2  
1857 2  
1858 2  
1859 2  
1860 2  
1861 2  
1862 2  
1863 2  
1864 2  
1865 2  
1866 2  
1867 2  
1868 2  
1869 2  
1870 2  
1871 2  
1872 2  
1873 2  
1874 2  
1875 2  
1876 2  
1877 2  
1878 2  
1879 2  
1880 2  
1881 2  
1882 2  
1883 2  
1884 2  
1885 2  
1886 2  
1887 2  
1888 2  
1889 2  
1890 2  
1891 2  
1892 2  
1893 2  
1894 2  
1895 2  
1896 2  
1897 2  
1898 2  
1899 2  
1900 2  
1901 2  
1902 2  
1903 2  
1904 2  
1905 2  
1906 2  
1907 2  
1908 2  
1909 2  
1910 2  
1911 2  
1912 2  
1913 2  
1914 2  
1915 2  
1916 2  
1917 2  
1918 2  
1919 2  
1920 2  
1921 2  
1922 2  
1923 2  
1924 2  
1925 2  
1926 2  
1927 2  
1928 2  
1929 2  
1930 2  
1931 2  
1932 2  
1933 2  
1934 2  
1935 2  
1936 2  
1937 2  
1938 2  
1939 2  
1940 2  
1941 2  
1942 2  
1943 2  
1944 2  
1945 2  
1946 2  
1947 2  
1948 2  
1949 2  
1950 2  
1951 2  
1952 2  
1953 2  
1954 2  
1955 2  
1956 2  
1957 2  
1958 2  
1959 2  
1960 2  
1961 2  
1962 2  
1963 2  
1964 2  
1965 2  
1966 2  
1967 2  
1968 2  
1969 2  
1970 2  
1971 2  
1972 2  
1973 2  
1974 2  
1975 2  
1976 2  
1977 2  
1978 2  
1979 2  
1980 2  
1981 2  
1982 2  
1983 2  
1984 2  
1985 2  
1986 2  
1987 2  
1988 2  
1989 2  
1990 2  
1991 2  
1992 2  
1993 2  
1994 2  
1995 2  
1996 2  
1997 2  
1998 2  
1999 2  
2000 2  
2001 2  
2002 2  
2003 2  
2004 2  
2005 2  
2006 2  
2007 2  
2008 2  
2009 2  
2010 2  
2011 2  
2012 2  
2013 2  
2014 2  
2015 2  
2016 2  
2017 2  
2018 2  
2019 2  
2020 2  
2021 2  
2022 2  
2023 2  
2024 2  
2025 2  
2026 2  
2027 2  
2028 2  
2029 2  
2030 2  
2031 2  
2032 2  
2033 2  
2034 2  
2035 2  
2036 2  
2037 2  
2038 2  
2039 2  
2040 2  
2041 2  
2042 2  
2043 2  
2044 2  
2045 2  
2046 2  
2047 2  
2048 2  
2049 2  
2050 2  
2051 2  
2052 2  
2053 2  
2054 2  
2055 2  
2056 2  
2057 2  
2058 2  
2059 2  
2060 2  
2061 2  
2062 2  
2063 2  
2064 2  
2065 2  
2066 2  
2067 2  
2068 2  
2069 2  
2070 2  
2071 2  
2072 2  
2073 2  
2074 2  
2075 2  
2076 2  
2077 2  
2078 2  
2079 2  
2080 2  
2081 2  
2082 2  
2083 2  
2084 2  
2085 2  
2086 2  
2087 2  
2088 2  
2089 2  
2090 2  
2091 2  
2092 2  
2093 2  
2094 2  
2095 2  
2096 2  
2097 2  
2098 2  
2099 2  
2100 2  
2101 2  
2102 2  
2103 2  
2104 2  
2105 2  
2106 2  
2107 2  
2108 2  
2109 2  
2110 2  
2111 2  
2112 2  
2113 2  
2114 2  
2115 2  
2116 2  
2117 2  
2118 2  
2119 2  
2120 2  
2121 2  
2122 2  
2123 2  
2124 2  
2125 2  
2126 2  
2127 2  
2128 2  
2129 2  
2130 2  
2131 2  
2132 2  
2133 2  
2134 2  
2135 2  
2136 2  
2137 2  
2138 2  
2139 2  
2140 2  
2141 2  
2142 2  
2143 2  
2144 2  
2145 2  
2146 2  
2147 2  
2148 2  
2149 2  
2150 2  
2151 2  
2152 2  
2153 2  
2154 2  
2155 2  
2156 2  
2157 2  
2158 2  
2159 2  
2160 2  
2161 2  
2162 2  
2163 2  
2164 2  
2165 2  
2166 2  
2167 2  
2168 2  
2169 2  
2170 2  
2171 2  
2172 2  
2173 2  
2174 2  
2175 2  
2176 2  
2177 2  
2178 2  
2179 2  
2180 2  
2181 2  
2182 2  
2183 2  
2184 2  
2185 2  
2186 2  
2187 2  
2188 2  
2189 2  
2190 2  
2191 2  
2192 2  
2193 2  
2194 2  
2195 2  
2196 2  
2197 2  
2198 2  
2199 2  
2200 2  
2201 2  
2202 2  
2203 2  
2204 2  
2205 2  
2206 2  
2207 2  
2208 2  
2209 2  
2210 2  
2211 2  
2212 2  
2213 2  
2214 2  
2215 2  
2216 2  
2217 2  
2218 2  
2219 2  
2220 2  
2221 2  
2222 2  
2223 2  
2224 2  
2225 2  
2226 2  
2227 2  
2228 2  
2229 2  
2230 2  
2231 2  
2232 2  
2233 2  
2234 2  
2235 2  
2236 2  
2237 2  
2238 2  
2239 2  
2240 2  
2241 2  
2242 2  
2243 2  
2244 2  
2245 2  
2246 2  
2247 2  
2248 2  
2249 2  
2250 2  
2251 2  
2252 2  
2253 2  
2254 2  
2255 2  
2256 2  
2257 2  
2258 2  
2259 2  
2260 2  
2261 2  
2262 2  
2263 2  
2264 2  
2265 2  
2266 2  
2267 2  
2268 2  
2269 2  
2270 2  
2271 2  
2272 2  
2273 2  
2274 2  
2275 2  
2276 2  
2277 2  
2278 2  
2279 2  
2280 2  
2281 2  
2282 2  
2283 2  
2284 2  
2285 2  
2286 2  
2287 2  
2288 2  
2289 2  
2290 2  
2291 2  
2292 2  
2293 2  
2294 2  
2295 2  
2296 2  
2297 2  
2298 2  
2299 2  
2300 2  
2301 2  
2302 2  
2303 2  
2304 2  
2305 2  
2306 2  
2307 2  
2308 2  
2309 2  
2310 2  
2311 2  
2312 2  
2313 2  
2314 2  
2315 2  
2316 2  
2317 2  
2318 2  
2319 2  
2320 2  
2321 2  
2322 2  
2323 2  
2324 2  
2325 2  
2326 2  
2327 2  
2328 2  
2329 2  
2330 2  
2331 2  
2332 2  
2333 2  
2334 2  
2335 2  
2336 2  
2337 2  
2338 2  
2339 2  
2340 2  
2341 2  
2342 2  
2343 2  
2344 2  
2345 2  
2346 2  
2347 2  
2348 2  
2349 2  
2350 2  
2351 2  
2352 2  
2353 2  
2354 2  
2355 2  
2356 2  
2357 2  
2358 2  
2359 2  
2360 2  
2361 2  
2362 2  
2363 2  
2364 2  
2365 2  
2366 2  
2367 2  
2368 2  
2369 2  
2370 2  
2371 2  
2372 2  
2373 2  
2374 2  
2375 2  
2376 2  
2377 2  
2378 2  
2379 2  
2380 2  
2381 2  
2382 2  
2383 2  
2384 2  
2385 2  
2386 2  
2387 2  
2388 2  
2389 2  
2390 2  
2391 2  
2392 2  
2393 2  
2394 2  
2395 2  
2396 2  
2397 2  
2398 2  
2399 2  
2400 2  
2401 2  
2402 2  
2403 2  
2404 2  
2405 2  
2406 2  
2407 2  
2408 2  
2409 2  
2410 2  
2411 2  
2412 2  
2413 2  
2414 2  
2415 2  
2416 2  
2417 2  
2418 2  
2419 2  
2420 2  
2421 2  
2422 2  
2423 2  
2424 2  
2425 2  
2426 2  
2427 2  
2428 2  
2429 2  
2430 2  
2431 2  
2432 2  
2433 2  
2434 2  
2435 2  
2436 2  
2437 2  
2438 2  
2439 2  
2440 2  
2441 2  
2442 2  
2443 2  
2444 2  
2445 2  
2446 2  
2447 2  
2448 2  
2449 2  
2450 2  
2451 2  
2452 2  
2453 2  
2454 2  
2455 2  
2456 2  
2457 2  
2458 2  
2459 2  
2460 2  
2461 2  
2462 2  
2463 2  
2464 2  
2465 2  
2466 2  
2467 2  
2468 2  
2469 2  
2470 2  
2471 2  
2472 2  
2473 2  
2474 2  
2475 2  
2476 2  
2477 2  
2478 2  
2479 2  
2480 2  
2481 2  
2482 2  
2483 2  
2484 2  
2485 2  
2486 2  
2487 2  
2488 2  
2489 2  
2490 2  
2491 2  
2492 2  
2493 2  
2494 2  
2495 2  
2496 2  
2497 2  
2498 2  
2499 2  
2500 2  
2501 2  
2502 2  
2503 2  
2504 2  
2505 2  
2506 2  
2507 2  
2508 2  
2509 2  
2510 2  
2511 2  
2512 2  
2513 2  
2514 2  
2515 2  
2516 2  
2517 2  
2518 2  
2519 2  
2520 2  
2521 2  
2522 2  
2523 2  
2524 2  
2525 2  
2526 2  
2527 2  
2528 2  
2529 2  
2530 2  
2531 2  
2532 2  
2533 2  
2534 2  
2535 2  
2536 2  
2537 2  
2538 2  
2539 2  
2540 2  
2541 2  
2542 2  
2543 2  
2544 2  
2545 2  
2546 2  
2547 2  
2548 2  
2549 2  
2550 2  
2551 2  
2552 2  
2553 2  
2554 2  
2555 2  
2556 2  
2557 2  
2558 2  
2559 2  
2560 2  
2561 2  
2562 2  
2563 2  
2564 2  
2565 2  
2566 2  
2567 2  
2568 2  
2569 2  
2570 2  
2571 2  
2572 2  
2573 2  
2574 2  
2575 2  
2576 2  
2577 2  
2578 2  
2579 2  
2580 2  
2581 2  
2582 2  
2583 2  
2584 2  
2585 2  
2586 2  
2587 2  
2588 2  
2589 2  
2590 2  
2591 2  
2592 2  
2593 2  
2594 2  
2595 2  
2596 2  
2597 2  
2598 2  
2599 2  
2600 2  
2601 2  
2602 2  
2603 2  
2604 2  
2605 2  
2606 2  
2607 2  
2608 2  
2609 2  
2610 2  
2611 2  
2612 2  
2613 2  
2614 2  
2615 2  
2616 2  
2617 2  
2618 2  
2619 2  
2620 2  
2621 2  
2622 2  
2623 2  
2624 2  
2625 2  
2626 2  
2627 2  
2628 2  
2629 2  
2630 2  
2631 2  
2632 2  
2633 2  
2634 2  
2635 2  
2636 2  
2637 2  
2638 2  
2639 2  
2640 2  
2641 2  
2642 2  
2643 2  
2644 2  
2645 2  
2646 2  
2647 2  
2648 2  
2649 2  
2650 2  
2651 2  
2652 2  
2653 2  
2654 2  
2655 2  
2656 2  
2657 2  
2658 2  
2659 2  
2660 2  
2661 2  
2662 2  
2663 2  
2664 2  
2665 2  
2666 2  
2667 2  
2668 2  
2669 2  
2670 2  
2671 2  
2672 2  
2673 2  
2674 2  
2675 2  
2676 2  
2677 2  
2678 2  
2679 2  
2680 2  
2681 2  
2682 2  
2683 2  
2684 2  
2685 2  
2686 2  
2687 2  
2688 2  
2689 2  
2690 2  
2691 2  
2692 2  
2693 2  
2694 2  
2695 2  
2696 2  
2697 2  
2698 2  
2699 2  
2700 2  
2701 2  
2702 2  
2703 2  
2704 2  
2705 2  
2706 2  
2707 2  
2708 2  
2709 2  
2710 2  
2711 2  
2712 2  
2713 2  
2714 2  
2715 2  
2716 2  
2717 2  
27
```

```

270 1857 3
271 1858
272 1859
273 1860
274 1861
275 1862
276 1863
277 1864
278 1865
279 1866
280 1867
281 1868
282 1869
283 1870
284 1871
285 1872
286 1873
287 1874
288 1875
289 1876
290 1877
291 1878
292 1879
293 1880
294 1881
295 1882
296 1883

    CHSMOVE (3, VT05_LINE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 3;
    END;

    [VT52]:
    BEGIN
    CHSMOVE (2, VT52_LINE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 2;
    END;

    [VT100]:
    BEGIN
    CHSMOVE (3, VT100_LINE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 3;
    END;

    [HARDCOPY, UNKNOWN, VTFOREIGN]:
    :

    [INRANGE, OUTRANGE]:
    RETURN 0;           ! should never get here

    TES:
    RETURN (SSS_NORMAL);

    END;                ! End of routine COB$SERASE_LINE_R2

```

00	00	1E	0003E	P.AAD:	.BLKB	2	
			00040		.BYTE	30, 0, 0	
			00043		.BLKB	1	
	4B	1B	00044	P.AAE:	.BYTE	27, 75	
			00046		.BLKB	2	
	4B	5B	1B	00048	P.AAF:	.BYTE	27, 91, 75

VT05_LINE=	P.AAD
VT52_LINE=	P.AAE
VT100_LINE=	P.AAF

001F	05	51	62	CO 00000 COB\$SERASE LINE_R2::		1851
	0016	00	50	ADDL2 (CUR_SIZE), FREE_ADDR		1853
		000E	CF 00003	CASEL TERM-TYPE, #0, #5		
		0028	00007 1\$:	.WORD	6S-1\$,-	
		0028	0000F		2S-1\$,-	
					3S-1\$,-	
					4S-1\$,-	
					6S-1\$,-	
					6S-1\$,-	
61	18	00	1E	11 00013	BRB	7S
		DD	AF	F0 00015 2\$:	INSV	VT05_LINE, #0, #24, (FREE_ADDR)
			0F	11 0001B	BRB	1857
		61	AF	B0 0001D 3\$:	MOVW	1858
		62	02	C0 00021	ADDL2	1863
			09	11 00024	BRB	1864
						1853

COB\$ESCAPE\_GEN COB\$ESCAPE GENERATOR - Escape sequence generat F 11  
1-003 COB\$SERASE\_LINE\_R2 - Create erase line sequence 16-Sep-1984 00:06:34 VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBESCGEN.B32;1

Page 10  
(4)

61	18	00	D4	AF	F0 00026 4\$:	INSV	VT100 LINE #0, #24, (FREE_ADDR)
62		03	C0	0002C 5\$:	ADDL2	#3, (CUR_SIZE)	
		50	01	D0 0002F 6\$:	MOVL	#1, R0	
				05 00032	RSB		
		50	D4	00033 7\$:	CLRL	R0	
				05 00035	RSB		

: 1869  
: 1870  
: 1881  
: 1883

: Routine Size: 54 bytes, Routine Base: \_COBSCODE + 0048

: 297 1884 1 !<BLF/PAGE>

```

299 1885 1 %SBTTL 'COB$SERASE PAGE R2 - Create erase page sequence'
300 1886 1 GLOBAL ROUTINE COB$SERASE PAGE_R2 (
301 1887 1 TERM_TYPE,
302 1888 1 BUFFER,
303 1889 1 CUR_SIZE
304 1890 1 ) : COB$SESC_R2_LNK =
305 1891 1 ++
306 1892 1 FUNCTIONAL DESCRIPTION:
307 1893 1
308 1894 1 This routine generates the escape sequence for erasing the
309 1895 1 page from the current cursor position to the end of the
310 1896 1 page. The sequence is appended into the output buffer.
311 1897 1
312 1898 1 CALLING SEQUENCE:
313 1899 1
314 1900 1 ret_status.wlc.v = COB$SERASE_PAGE_R2 (TERM_TYPE.rl.v,
315 1901 1 BUFFER.mt.r, CUR_SIZE.ml.r)
316 1902 1
317 1903 1 FORMAL PARAMETERS:
318 1904 1
319 1905 1 TERM_TYPE.rl.v terminal type
320 1906 1 BUFFER.mt.r addr of buffer
321 1907 1 CUR_SIZE.ml.r # bytes currently in buffer
322 1908 1
323 1909 1 IMPLICIT INPUTS:
324 1910 1
325 1911 1 NONE
326 1912 1
327 1913 1 IMPLICIT OUTPUTS:
328 1914 1
329 1915 1 NONE
330 1916 1
331 1917 1 COMPLETION STATUS:
332 1918 1
333 1919 1
334 1920 1 SIDE EFFECTS:
335 1921 1
336 1922 1 NONE
337 1923 1 --
338 1924 1
339 1925 2 BEGIN
340 1926 2
341 1927 2 LOCAL
342 1928 2 FREE_ADDR; ! addr of next free byte in buffer
343 1929 2
344 1930 2 BIND
345 1931 2 VT05_ERASE = UPLIT (BYTE (VT05_EOS, NULL, NULL)),
346 1932 2 VT52_ERASE = UPLIT (BYTE (ESC, VT52_EOS)),
347 1933 2 VT100_ERASE = UPLIT (BYTE (ESC, LB, VT100_EOS));
348 1934 2
349 1935 2 FREE_ADDR = .BUFFER + ..CUR_SIZE;
350 1936 2
351 1937 2 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
352 1938 2 SET
353 1939 2 [VT05]:
354 1940 2 BEGIN
355 1941 3 CHSMOVE (3, VT05_ERASE, .FREE_ADDR);

```

```

356 1942 .CUR_SIZE = ..[CUR_SIZE + 3;
357 1943 END;
358 1944
359 1945 [VT52]:
360 1946 BEGIN
361 1947 CHSMOVE (2, VT52_ERASE, .FREE_ADDR);
362 1948 .CUR_SIZE = ..[CUR_SIZE + 2;
363 1949 END;
364 1950
365 1951 [VT100]:
366 1952 BEGIN
367 1953 CHSMOVE (3, VT100_ERASE, .FREE_ADDR);
368 1954 .CUR_SIZE = ..[CUR_SIZE + 3;
369 1955 END;
370 1956
371 1957 [HARDCOPY, UNKNOWN, VTFOREIGN]:
372 1958 :
373 1959
374 1960 [INRANGE, OUTRANGE]:
375 1961 RETURN 0; ! should never get here
376 1962
377 1963 TES;
378 1964
379 1965 RETURN (SSS_NORMAL);
380 1966
381 1967 1 END; ! End of routine COBSSERASE_PAGE_R2
    
```

00	00	1F	00081	P.AAG:	.BLKB	3
			00084	.BYTE		31, 0, 0
			00087	.BLKB	1	
			00088	P.AAH:	.BYTE	27, 74
			0008A	.BLKB	2	
			0008C	P.AAI:	.BYTE	27, 91, 74
				VT05_ERASE=		P.AAG
				VT52_ERASE=		P.AAH
				VT100_ERASE=		P.AAI

001F	0016	05	51	62	CO 00000 COBSSERASE PAGE_R2::	1935 1937
			00	50	ADDL2 (CUR_SIZE), FREE_ADDR	
			000E	CF 00003	CASEL TERM-TYPE, #0, #5	
			0028	00007 1\$: .WORD	6\$-1\$,- 2\$-1\$,- 3\$-1\$,- 4\$-1\$,- 6\$-1\$,- 6\$-1\$	
61	18	00 DD 1E 11 00013 2\$: BRB AF F0 00015 2\$: INSV OF 11 0001B 2\$: BRB 58	1961 1941 1942 1947 1948 1937 1953			
		61 D9 AF B0 0001D 3\$: MOVW 62 02 C0 00021 3\$: ADDL2 09 11 00024 3\$: BRB 68				
		VT05_ERASE, #0, #24, (FREE_ADDR) VT52_ERASE, (FREE_ADDR) #2, (CUR_SIZE)				
		VT100_ERASE, #0, #24, (FREE_ADDR)				

COB\$ESCAPE\_GEN COB\$ESCAPE GENERATOR - Escape sequence generat 16-Sep-1984 00:06:34  
1-003 COB\$ERASE\_PAGE\_R2 - Create erase page sequence 14-Sep-1984 12:10:44 VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBESCGEN.B32:1

Page 13  
(5)

62 03 C0 0002C 58: ADDL2 #3, (CUR\_SIZE)  
50 01 D0 0002F 68: MOVL #1, R0  
50 05 00032 RSB  
50 D4 00033 78: CLRL R0  
05 00035 RSB

; 1954  
; 1965  
; 1967

; Routine Size: 54 bytes, Routine Base: \_COB\$CODE + 008F

; 382 1968 1 !<BLF/PAGE>

384 1969 1 ZSBTTL 'COB\$ERASE\_WHOLE\_LINE\_R2 - Create erase whole line sequence'  
385 1970 1 GLOBAL ROUTINE COB\$ERASE\_WHOLE\_LINE\_R2 (   
386 1971 1 TERM\_TYPE,  
387 1972 1 BUFFER,  
388 1973 1 CUR\_SIZE  
389 1974 1 ) : COB\$ESC\_R2\_LNK =  
390 1975 1 ++  
391 1976 1 FUNCTIONAL DESCRIPTION:  
392 1977 1  
393 1978 1 This routine generates the escape sequence to erase the entire  
394 1979 1 line containing the current cursor position. The string is  
395 1980 1 appended into the output buffer.  
396 1981 1  
397 1982 1 Notice that only VT100s have the ability to erase an entire  
398 1983 1 line regardless of whether the cursor is at the beginning  
399 1984 1 of that line. Most terminals can only erase from the cursor  
400 1985 1 to the end of line.  
401 1986 1  
402 1987 1 CALLING SEQUENCE:  
403 1988 1  
404 1989 1 ret\_status.wlc.v = COB\$ERASE\_WHOLE\_LINE\_R2 (TERM\_TYPE.rl.v,  
405 1990 1 BUFFER.ml.r,  
406 1991 1 CUR\_SIZE.ml.r)  
407 1992 1  
408 1993 1 FORMAL PARAMETERS:  
409 1994 1  
410 1995 1 TERM\_TYPE.rl.v terminal type  
411 1996 1 BUFFER.ml.r addr of buffer  
412 1997 1 CUR\_SIZE.ml.r # bytes currently in buffer  
413 1998 1  
414 1999 1 IMPLICIT INPUTS:  
415 2000 1 NONE  
416 2001 1  
417 2002 1  
418 2003 1 IMPLICIT OUTPUTS:  
419 2004 1 NONE  
420 2005 1  
421 2006 1  
422 2007 1 COMPLETION STATUS:  
423 2008 1  
424 2009 1  
425 2010 1 SIDE EFFECTS:  
426 2011 1 NONE  
427 2012 1  
428 2013 1 --  
429 2014 1  
430 2015 2 BEGIN  
431 2016 2  
432 2017 2  
433 2018 2 LOCAL FREE\_ADDR; ! addr of next free byte in buffer  
434 2019 2  
435 2020 2 BIND  
436 2021 2 VT05\_LINE = UPLIT (BYTE (VT05\_EOL, NULL, NULL)),  
437 2022 2 VT52\_LINE = UPLIT (BYTE (ESC, VT52\_EOL)),  
438 2023 2 VT100\_WHOLE\_LINE = UPLIT (BYTE (ESC, LB, TWO, VT100\_EOL));  
439 2024 2  
440 2025 2 ! EE\_ADDR = .BUFFER + ..CUR\_SIZE;

```

441 2026 2
442 2027
443 2028
444 2029
445 2030
446 2031
447 2032
448 2033
449 2034
450 2035
451 2036
452 2037
453 2038
454 2039
455 2040
456 2041
457 2042
458 2043
459 2044
460 2045
461 2046
462 2047
463 2048
464 2049
465 2050
466 2051
467 2052
468 2053
469 2054
470 2055
471 2056
472 2057 1

CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
SET
    [VT05]:
        BEGIN
            CHSMOVE (3, VT05_LINE, .FREE_ADDR);
            .CUR_SIZE = ..CUR_SIZE + 3;
        END;

    [VT52]:
        BEGIN
            CHSMOVE (2, VT52_LINE, .FREE_ADDR);
            .CUR_SIZE = ..CUR_SIZE + 2;
        END;

    [VT100]:
        BEGIN
            CHSMOVE (4, VT100_WHOLE_LINE, .FREE_ADDR);
            .CUR_SIZE = ..CUR_SIZE + 4;
        END;

    [HARDCOPY, UNKNOWN, VTFOREIGN]:
        :
    [INRANGE, OUTRANGE]:
        RETURN 0;                                ! should never get here

    TES;

    RETURN (SS$_NORMAL);

END:                                         ! End of routine COB$SERASE_WHOLE_LINE_R2

```

00	00	1E	000C5	000C8	P.AAJ:	.BLKB	3				
				000CB		.BYTE	30, 0, 0				
				4B	1B	000CC	P.AAK:	.BLKB	1		
						000CE		.BYTE	27, 75		
				4B	32	5B	1B	000D0	P.AAL:	.BLKB	2
										.BYTE	27, 91, 50, 75
											VT05_LINE= P.AAJ
											VT52_LINE= P.AAK
											VT100_WHOLE_LINE= P.AAL

0022	0019	05	51	62	CO 00000 COB\$SERASE WHOLE LINE R2::
		00		50	ADDL2 ?CUR_SIZE), .FREE_ADDR
		000E		CF 00003	CASEL TERM_TYPE, #0, #5
		0029		00007 1\$:	.WORD 58-18,-
		0029		0000F	28-18,-
					38-18,-
					48-18,-
					58-18,-
					58-18
			1F 11 00013	BRB	68

: 2025  
 : 2027

: 2051

L 11  
COB\$ESCAPE\_GEN COB\$ESCAPE\_GENERATOR - Escape sequence generat 16-Sep-1984 00:06:34  
1-003 COB\$ERASE\_WHOLE\_LINE\_R2 - Create erase whole 14-Sep-1984 12:10:44 VAX-11 Blfse-32 V4.0-742  
[COBRTL.SRC]COBESCGEN.B32:1

Page 16  
(6)

61	18	00	DC	AF	F0	00015	28:	INSV	VT05 LINE, #0, #24, (FREE_ADDR)
62		03	CO	0001B				ADDL2	#3, TCUR_SIZE
		10	11	0001E				BRB	55
61		D5	AF	BO	00020	38:	MOVW	VT52 LINE, (FREE_ADDR)	
62		02	CO	00024				ADDL2	#2, TCUR_SIZE
61		07	11	00027				BRB	55
61		D0	AF	DO	00029	48:	MOVL	VT100 WHOLE LINE, (FREE_ADDR)	
62		04	CO	0002D				ADDL2	#4, (CUR_SIZE)
50		01	DO	00030	58:			MOVL	#1, R0
				05	00033			RSB	
		50	D4	00034	68:			CLRL	R0
				05	00036			RSB	

: Routine Size: 55 bytes. Routine Base: \_COB\$CODE + 00D4

: 473 2058 1 !<BLF/PAGE>

```

475 2059 1 ZSBTTL 'COB$SERASE WHOLE PAGE R2 - Create erase whole page sequence'
476 2060 1 GLOBAL ROUTINE COB$SERASE_WHOLE_PAGE_R2 (
477 2061 1 TERM_TYPE,
478 2062 1 BUFFER,
479 2063 1 CUR_SIZE
480 2064 1 ) : COB$SESC_R2_LNK =
481 2065 1 ++
482 2066 1 FUNCTIONAL DESCRIPTION:
483 2067 1
484 2068 1 This routine generates the escape sequence to erase the
485 2069 1 whole page regardless of cursor position. The string is appended
486 2070 1 into the output buffer.
487 2071 1
488 2072 1 CALLING SEQUENCE:
489 2073 1
490 2074 1     ret_status.wlc.v = COB$SERASE_WHOLE_PAGE_R2 (TERM_TYPE.rl.v,
491 2075 1                               BUFFER.mt.r,
492 2076 1                               CUR_SIZE.ml.r)
493 2077 1
494 2078 1 FORMAL PARAMETERS:
495 2079 1
496 2080 1     TERM_TYPE.rl.v          terminal type
497 2081 1     BUFFER.mt.r          addr of buffer
498 2082 1     CUR_SIZE.ml.r        # bytes currently in buffer
499 2083 1
500 2084 1 IMPLICIT INPUTS:
501 2085 1
502 2086 1     NONE
503 2087 1
504 2088 1 IMPLICIT OUTPUTS:
505 2089 1
506 2090 1     NONE
507 2091 1
508 2092 1 COMPLETION STATUS:
509 2093 1
510 2094 1
511 2095 1
512 2096 1
513 2097 1
514 2098 1
515 2099 1
516 2100 2
517 2101 2
518 2102 2
519 2103 2
520 2104 2
521 2105 2
522 2106 2
523 2107 2
524 2108 2
525 2109 2
526 2110 2
527 2111 2
528 2112 2
529 2113 2
530 2114 2
531 2115 2

    SIDE EFFECTS:
    NONE
    --
    BEGIN
    LOCAL
        FREE_ADDR;                      ! addr of next free byte in buffer
    LITERAL
        LINE1 = 32;                      ! 1 + 31 bias
        COL1 = 32;                       ! 1 + 31 bias
    BIND
        VT05_ERASE = UPLIT (BYTE (VT05_EOS, NULL, NULL)),
        VT52_ERASE = UPLIT (BYTE (ESC, VT52_SC, LINE1, COL1,
                                  ESC, VT52_EOS)),
        VT100_ERASE_WHOLE = UPLIT (BYTE (ESC, LB, TWO, VT100_EOS));
    FREE_ADDR = .BUFFER + ..CUR_SIZE;

```

```

532      2116 2
533      2117 2
534      2118 2
535      2119 2
536      2120 2
537      2121 2
538      2122 2
539      2123 2
540      2124 2
541      2125 2
542      2126 2
543      2127 2
544      2128 2
545      2129 2
546      2130 2
547      2131 2
548      2132 2
549      2133 2
550      2134 2
551      2135 2
552      2136 2
553      2137 2
554      2138 2
555      2139 2
556      2140 2
557      2141 2
558      2142 2
559      2143 2
560      2144 2
561      2145 2
562      2146 2
563      2147 2
564      2148 2
565      2149 2
566      2150 2
567      2151 2
568      2152 1

CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
SET
[V1100]:
BEGIN
    CHSMOVE (4, VT100_ERASE_WHOLE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 4;
END;

[V152]:
BEGIN
    +
    There is no sequence to erase the screen and leave the
    cursor where it was, so on a VT52 we have to settle for
    setting the cursor to 1,1 and erasing to the end of screen.
    -
    CHSMOVE (6, VT52_ERASE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 6;
END;

[VT05]:
BEGIN
    CHSMOVE (3, VT05_ERASE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 3;
END;

[HARDCOPY, UNKNOWN, VTFOREIGN]:
:
[INRANGE, OUTRANGE]:
    RETURN 0;           ! should never get here

TES:
RETURN (SSS_NORMAL);
END;                      ! End of routine COB$ERASE_WHOLE_PAGE_R2

```

00	00	1F	0010B	.BLKB	1
4A	1B	20	0010C P.AAM:	.BYTE	31, 0, 0
		59	0010F	.BLKB	1
		1B	00110 P.AAN:	.BYTE	27, 89, 32, 32, 27, 74
			00116	.BLKB	2
			00118 P.AAO:	.BYTE	27, 91, 50, 74
VT05_ERASE= P.AAM					
VT52_ERASE= P.AAN					
VT100_ERASE_WHOLE= P.AAO					

57	00	00F8	8F BB 00000 COB\$ERASE_WHOLE PAGE R2::	PUSHR	3^M<R3,R4,R5,R6,R7>	:	2060
51	56	52 D0 00004		MOVL	R2, R6	:	2115
05	00	66 C1 00007		ADDL3	(CUR_SIZE), BUFFER, FREE_ADDR	:	2117
		50 CF 0000B		CASEL	TERM_TYPE, #0, #5		

; Routine Size: 69 bytes, Routine Base: \_COB\$CODE + 011C

: 569 2153 1 !<BLF/PAGE>

```

571 2154 1 %SBTTL 'COB$SET_ATTRIBUTES - Create set attributes sequence'
572 2155 1 GLOBAL ROUTINE COB$SET_ATTRIBUTES (
573 2156 1 TERM_TYPE,
574 2157 1 IN_TEXT,
575 2158 1 IN_LEN,
576 2159 1 FLAGS,
577 2160 1 OUT_BUF,
578 2161 1 OUT_LEN
579 2162 1 ) =
580 2163 1 ++
581 2164 1 FUNCTIONAL DESCRIPTION:
582 2165 1
583 2166 1 This routine generates the escape sequence turning on
584 2167 1 attributes such as bolding and blinking. The attribute
585 2168 1 sequence is placed in the output buffer, the input text
586 2169 1 is copied over, and then the sequence to turn off graphics
587 2170 1 is appended.
588 2171 1
589 2172 1 CALLING SEQUENCE:
590 2173 1
591 2174 1 ret_status.wlc.v = COB$SET_ATTRIBUTES (TERM_TYPE.rl.v, IN_TEXT.rt.r,
592 2175 1 IN_LEN.rl.v, FLAGS.rl.v,
593 2176 1 OUT_BUF.mt.r, OUT_LEN.ml.r)
594 2177 1
595 2178 1 FORMAL PARAMETERS:
596 2179 1
597 2180 1 TERM_TYPE.rl.v terminal type
598 2181 1 IN_TEXT.rt.dx descriptor of text which will have attr on
599 2182 1 IN_LEN.rl.v length of caller's text
600 2183 1 FLAGS.rl.v flags specifying which attributes to turn on
601 2184 1 OUT_BUF.mt.r addr of output buffer
602 2185 1 OUT_LEN.ml.r # bytes in output buffer, includes attributes,
603 2186 1 caller's text, & turn off graphic rendition
604 2187 1
605 2188 1
606 2189 1
607 2190 1
608 2191 1
609 2192 1
610 2193 1
611 2194 1
612 2195 1
613 2196 1
614 2197 1
615 2198 1
616 2199 1
617 2200 1
618 2201 1
619 2202 1
620 2203 1
621 2204 2
622 2205 2
623 2206 2
624 2207 2
625 2208 2
626 2209 2
627 2210 2

```

IMPLICIT INPUTS:  
NONE

IMPLICIT OUTPUTS:  
NONE

COMPLETION STATUS:

SIDE EFFECTS:  
NONE

--

BEGIN

LOCAL  
FREE\_ADDR;

MACRO  
VT100\_OFF = %STRING (%CHAR (ESC), %CHAR (LB), '0m')%;

```

628      2211 2      FREE_ADDR = .OUT_BUF + ..OUT_LEN;           ! init to first free byte
629      2212 2
630      2213 2
631      2214 2      CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
632      2215 2      SET
633      2216 2      [HARDCOPY, UNKNOWN, VT05, VT52, VTFOREIGN]:
634      2217 2      BEGIN
635      2218 2      !+
636      2219 2      | Renditions not supported on these devices. Just
637      2220 2      | copy the text into the output buffer and return.
638      2221 2
639      2222 2      CHSMOVE (.IN_LEN, .IN_TEXT, .FREE_ADDR);
640      2223 2      .OUT_LEN = ..OUT_LEN + .IN_LEN;
641      2224 2      RETURN (SSS_NORMAL);
642      2225 2      END;
643      2226 2
644      2227 2      [INRANGE, OUTRANGE]:
645      2228 2      RETURN 0;                      ! error
646      2229 2
647      2230 2      [VT100]:
648      2231 2      BEGIN
649      2232 2      IF .FLAGS <0,4> EQL 0
650      2233 2      THEN
651      2234 2      BEGIN
652      2235 2      | no attr requested
653      2236 2      CHSMOVE (.IN_LEN, .IN_TEXT, .FREE_ADDR);
654      2237 2      .OUT_LEN = ..OUT_LEN + .IN_LEN;
655      2238 2      RETURN (SSS_NORMAL);
656      2239 2      END;
657      2240 2      !+
658      2241 2      | For each attribute bit set in flags, copy
659      2242 2      | the appropriate ASCII graphic rendition byte
660      2243 2      | followed by a ';' into the output buffer.
661      2244 2      | Note use of autoincrementing.
662      2245 2
663      2246 2      CHSWCHAR_A (ESC, FREE_ADDR);
664      2247 2      CHSWCHAR_A (LB, FREE_ADDR);
665      2248 2      INCR I FROM 0 TO 3
666      2249 2      DO
667      2250 2      BEGIN
668      2251 2      | build attribute string
669      2252 2      BIND
670      2253 2      ATTRTABL = UPLIT (BYTE ('1754')) : VECTOR [4, BYTE];
671      2254 2      IF .FLAGS <.I, 1>
672      2255 2      THEN
673      2256 2      BEGIN
674      2257 2      CHSWCHAR_A (.ATTRTABL [.I], FREE_ADDR);
675      2258 2      CHSWCHAR_A (XC';', FREE_ADDR);
676      2259 2      .OUT_LEN = ..OUT_LEN + 2; ! keep updating length
677      2260 2      END;
678      2261 2
679      2262 2
680      2263 2      !+
681      2264 2      | When we fall out of above loop we have deposited
682      2265 2      | an extra ';' at the end of the buffer. Back up
683      2266 2      | FREE_ADDR and write VT100_SGR on top of it.
684      2267 2      FREE_ADDR = .FREE_ADDR - 1;

```

```

685 2268 3 CH$UCHAR_A (VT100_SGR, FREE_ADDR);
686 2269 END;
687 2270 TES:
688 2271
689 2272
690 2273
691 2274
692 2275
693 2276
694 2277
695 2278
696 2279
697 2280
698 2281
699 2282
700 2283
701 2284
702 2285
703 2286
704 2287
705 2288
706 2289
707 2290 1

        If we get here, the appropriate graphic rendition string has
        been moved to the output buffer. Now copy the user's text over.

        FREE_ADDR = CHSMOVE (.IN_LEN, .IN_TEXT, .FREE_ADDR);

        Append in sequence to turn off graphic rendition.

        CHSMOVE (%CHARCOUNT (VT100_OFF), UPLIT (BYTE (VT100_OFF)), .FREE_ADDR);

        Set the output length and exit.

        .OUT_LEN = ..OUT_LEN + .IN_LEN + 6; ! add length of caller's text &
        ! turn on/off graphic rendition

        RETURN (SSS_NORMAL);

        END: ! End of routine COB$SET_ATTRIBUTES
    
```

00161 34 35 37 31 00164 P.AAP: .BLKB 3  
 6D 30 5B 1B 00164 P.AAQ: .ASCII \1754\  
 00168 P.AAQ: .ASCII <27>\[0m\]

ATTRTABL= P.AAP

000E	0014	57 05 14	56 00 0014 0014	18 04	00FC AC 00002 66 C1 00006 0004 AC CF 0000B 0014 00010 0014 0014 00018	0001C 93 0001E 28: 18:	.ENTRY MOVL ADDL3 CASEL .WORD	COB\$SET_ATTRIBUTES, Save R2,R3,R4,R5,R6,R7 OUT LEN, R6 (R6), OUT BUF, FREE_ADDR TERM TYPE, #0, #5 3\$-1\$,- 3\$-1\$,- 3\$-1\$,- 2\$-1\$,- 3\$-1\$,- 3\$-1\$,-	2155 2212 2214
		67	08	0F	10 AC 93 00022 0C AC 28 00024 66 0C AC C0 0002A 37 11 0002E	4D 11 0001C 28: 0C 12 00022 38: 38: MOVC3 ADDL2	BRB BITB BNEQ 4\$ IN_LEN, @IN_TEXT, (FREE_ADDR) IN_LEN, (R6) 7\$	2228 2232 2235 2236 2237	
		08	10	87	5B1B 8F 80 00030 50 D4 00035 B8 AF 40 90 0003C 87 3B 90 00041 66 02 C0 00044	50 E1 00037 48: 58: MOVW CLRL BBC MOVW MOVW ADDL2	#23323, (FREE_ADDR)+ I, FLAGS, 6\$ ATTRTABL[I], (FREE_ADDR)+ #59, (FREE_ADDR)+ #2, (R6) #3, I, 5\$	2245 2247 2253 2256 2257 2258 2247	
		EC		77	6D 8F 90 0004B	6D 03 F3 00047 68:	AOBLEQ MOVW	#109, -(FREE_ADDR)	2268

COB\$ESCAPE\_GEN COB\$ESCAPE\_GENERATOR - Escape sequence generat F 12  
1-003 COB\$SET\_ATTRIBUTES - Create set attributes seq 16-Sep-1984 00:06:34 14-Sep-1984 12:10:44 VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBESCGEN.B32;1

Page 23  
(8)

67	08	BC	0C	57	D6	0004F	INCL	FREE ADDR	2276	
		57		53	D0	00051	MOVC3	IN-LEN, @IN TEXT, (FREE_ADDR)		
	50	67	9F	AF	D0	00057	MOVL	R3, FREE ADDR	2281	
		66	0C	AC	C1	0005A	MOVL	P, @A0, (FREE_ADDR)	2286	
		66	06	A0	9E	00063	ADDL3	IN LEN, (R6), R0		
		50		01	D0	00067	78:	MOVAB	6(R0) (R6)	2288
					04	0006A	50	MOVL	#1, R0	
					04	0006B	88:	RET	RET	2290
								CLRL	R0	

: Routine Size: 110 bytes. Routine Base: \_COB\$CODE + 016C

: 708 2291 1 !<BLF/PAGE>

```

710      2292 1 %SBTTL 'COB$SET_ATTRIBUTES_ONLY - Create only set attributes sequence'
711      2293 1 GLOBAL ROUTINE COB$SET_ATTRIBUTES_ONLY (
712      2294 1     TERM_TYPE,
713      2295 1     FLAGS,
714      2296 1     PREFIX_BUF,
715      2297 1     P_PREFIX_LEN,
716      2298 1     SUFFIX_BUF,
717      2299 1     P_SUFFIX_LEN
718      2300 1     ) =
719      2301 1
720      2302 1     ++
721      2303 1     FUNCTIONAL DESCRIPTION:
722      2304 1
723      2305 1     This routine generates the escape sequences turning on and off
724      2306 1     attributes such as bolding and blinking. These attribute
725      2307 1     sequences are placed in two buffers supplied by the caller.
726      2308 1     No input text is specified.
727      2309 1
728      2310 1     CALLING SEQUENCE:
729      2311 1
730      2312 1     ret_status.wlc.v = COB$SET_ATTRIBUTES (TERM_TYPE.rl.v,
731      2313 1     FLAGS.rl.v,
732      2314 1     PREFIX_BUF.mt.r,
733      2315 1     P_PREFIX_LEN.ml.r,
734      2316 1     SUFFIX_BUF.mt.r,
735      2317 1     P_SUFFIX_LEN.ml.r)
736      2318 1
737      2319 1
738      2320 1     FORMAL PARAMETERS:
739      2321 1
740      2322 1     TERM_TYPE.rl.v
741      2323 1     FLAGS.rl.v
742      2324 1     PREFIX_BUF.mt.r
743      2325 1     P_PREFIX_LEN.ml.r
744      2326 1     SUFFIX_BUF.mt.r
745      2327 1     P_SUFFIX_LEN.ml.r
746      2328 1
747      2329 1     terminal type
748      2330 1     flags specifying which attributes to turn on
749      2331 1     addr of output buffer to receive prefix string
750      2332 1     # bytes in already in prefix buffer
751      2333 1     gets updated to include size of prefix
752      2334 1     addr of output buffer to receive suffix string
753      2335 1     # bytes in already in suffix buffer
754      2336 1     gets updated to include size of suffix
755      2337 1
756      2338 1
757      2339 1
758      2340 1
759      2341 1
760      2342 1
761      2343 1     COMPLETION STATUS:
762      2344 1
763      2345 1     SIDE EFFECTS:
764      2346 1
765      2347 1     NONE
766      2348 1
767      2349 1
768      2350 1
769      2351 1
770      2352 1
771      2353 1
772      2354 1
773      2355 1
774      2356 1
775      2357 1
776      2358 1
777      2359 1
778      2360 1
779      2361 1
780      2362 1
781      2363 1
782      2364 1
783      2365 1
784      2366 1
785      2367 1
786      2368 1
787      2369 1
788      2370 1
789      2371 1
790      2372 1
791      2373 1
792      2374 1
793      2375 1
794      2376 1
795      2377 1
796      2378 1
797      2379 1
798      2380 1
799      2381 1
800      2382 1
801      2383 1
802      2384 1
803      2385 1
804      2386 1
805      2387 1
806      2388 1
807      2389 1
808      2390 1
809      2391 1
810      2392 1
811      2393 1
812      2394 1
813      2395 1
814      2396 1
815      2397 1
816      2398 1
817      2399 1
818      2400 1
819      2401 1
820      2402 1
821      2403 1
822      2404 1
823      2405 1
824      2406 1
825      2407 1
826      2408 1
827      2409 1
828      2410 1
829      2411 1
830      2412 1
831      2413 1
832      2414 1
833      2415 1
834      2416 1
835      2417 1
836      2418 1
837      2419 1
838      2420 1
839      2421 1
840      2422 1
841      2423 1
842      2424 1
843      2425 1
844      2426 1
845      2427 1
846      2428 1
847      2429 1
848      2430 1
849      2431 1
850      2432 1
851      2433 1
852      2434 1
853      2435 1
854      2436 1
855      2437 1
856      2438 1
857      2439 1
858      2440 1
859      2441 1
860      2442 1
861      2443 1
862      2444 1
863      2445 1
864      2446 1
865      2447 1
866      2448 1
867      2449 1
868      2450 1
869      2451 1
870      2452 1
871      2453 1
872      2454 1
873      2455 1
874      2456 1
875      2457 1
876      2458 1
877      2459 1
878      2460 1
879      2461 1
880      2462 1
881      2463 1
882      2464 1
883      2465 1
884      2466 1
885      2467 1
886      2468 1
887      2469 1
888      2470 1
889      2471 1
890      2472 1
891      2473 1
892      2474 1
893      2475 1
894      2476 1
895      2477 1
896      2478 1
897      2479 1
898      2480 1
899      2481 1
900      2482 1
901      2483 1
902      2484 1
903      2485 1
904      2486 1
905      2487 1
906      2488 1
907      2489 1
908      2490 1
909      2491 1
910      2492 1
911      2493 1
912      2494 1
913      2495 1
914      2496 1
915      2497 1
916      2498 1
917      2499 1
918      2500 1
919      2501 1
920      2502 1
921      2503 1
922      2504 1
923      2505 1
924      2506 1
925      2507 1
926      2508 1
927      2509 1
928      2510 1
929      2511 1
930      2512 1
931      2513 1
932      2514 1
933      2515 1
934      2516 1
935      2517 1
936      2518 1
937      2519 1
938      2520 1
939      2521 1
940      2522 1
941      2523 1
942      2524 1
943      2525 1
944      2526 1
945      2527 1
946      2528 1
947      2529 1
948      2530 1
949      2531 1
950      2532 1
951      2533 1
952      2534 1
953      2535 1
954      2536 1
955      2537 1
956      2538 1
957      2539 1
958      2540 1
959      2541 1
960      2542 1
961      2543 1
962      2544 1
963      2545 1
964      2546 1
965      2547 1
966      2548 1
967      2549 1
968      2550 1
969      2551 1
970      2552 1
971      2553 1
972      2554 1
973      2555 1
974      2556 1
975      2557 1
976      2558 1
977      2559 1
978      2560 1
979      2561 1
980      2562 1
981      2563 1
982      2564 1
983      2565 1
984      2566 1
985      2567 1
986      2568 1
987      2569 1
988      2570 1
989      2571 1
990      2572 1
991      2573 1
992      2574 1
993      2575 1
994      2576 1
995      2577 1
996      2578 1
997      2579 1
998      2580 1
999      2581 1
1000     2582 1
1001     2583 1
1002     2584 1
1003     2585 1
1004     2586 1
1005     2587 1
1006     2588 1
1007     2589 1
1008     2590 1
1009     2591 1
1010     2592 1
1011     2593 1
1012     2594 1
1013     2595 1
1014     2596 1
1015     2597 1
1016     2598 1
1017     2599 1
1018     2600 1
1019     2601 1
1020     2602 1
1021     2603 1
1022     2604 1
1023     2605 1
1024     2606 1
1025     2607 1
1026     2608 1
1027     2609 1
1028     2610 1
1029     2611 1
1030     2612 1
1031     2613 1
1032     2614 1
1033     2615 1
1034     2616 1
1035     2617 1
1036     2618 1
1037     2619 1
1038     2620 1
1039     2621 1
1040     2622 1
1041     2623 1
1042     2624 1
1043     2625 1
1044     2626 1
1045     2627 1
1046     2628 1
1047     2629 1
1048     2630 1
1049     2631 1
1050     2632 1
1051     2633 1
1052     2634 1
1053     2635 1
1054     2636 1
1055     2637 1
1056     2638 1
1057     2639 1
1058     2640 1
1059     2641 1
1060     2642 1
1061     2643 1
1062     2644 1
1063     2645 1
1064     2646 1
1065     2647 1
1066     2648 1
1067     2649 1
1068     2650 1
1069     2651 1
1070     2652 1
1071     2653 1
1072     2654 1
1073     2655 1
1074     2656 1
1075     2657 1
1076     2658 1
1077     2659 1
1078     2660 1
1079     2661 1
1080     2662 1
1081     2663 1
1082     2664 1
1083     2665 1
1084     2666 1
1085     2667 1
1086     2668 1
1087     2669 1
1088     2670 1
1089     2671 1
1090     2672 1
1091     2673 1
1092     2674 1
1093     2675 1
1094     2676 1
1095     2677 1
1096     2678 1
1097     2679 1
1098     2680 1
1099     2681 1
1100     2682 1
1101     2683 1
1102     2684 1
1103     2685 1
1104     2686 1
1105     2687 1
1106     2688 1
1107     2689 1
1108     2690 1
1109     2691 1
1110     2692 1
1111     2693 1
1112     2694 1
1113     2695 1
1114     2696 1
1115     2697 1
1116     2698 1
1117     2699 1
1118     2700 1
1119     2701 1
1120     2702 1
1121     2703 1
1122     2704 1
1123     2705 1
1124     2706 1
1125     2707 1
1126     2708 1
1127     2709 1
1128     2710 1
1129     2711 1
1130     2712 1
1131     2713 1
1132     2714 1
1133     2715 1
1134     2716 1
1135     2717 1
1136     2718 1
1137     2719 1
1138     2720 1
1139     2721 1
1140     2722 1
1141     2723 1
1142     2724 1
1143     2725 1
1144     2726 1
1145     2727 1
1146     2728 1
1147     2729 1
1148     2730 1
1149     2731 1
1150     2732 1
1151     2733 1
1152     2734 1
1153     2735 1
1154     2736 1
1155     2737 1
1156     2738 1
1157     2739 1
1158     2740 1
1159     2741 1
1160     2742 1
1161     2743 1
1162     2744 1
1163     2745 1
1164     2746 1
1165     2747 1
1166     2748 1
1167     2749 1
1168     2750 1
1169     2751 1
1170     2752 1
1171     2753 1
1172     2754 1
1173     2755 1
1174     2756 1
1175     2757 1
1176     2758 1
1177     2759 1
1178     2760 1
1179     2761 1
1180     2762 1
1181     2763 1
1182     2764 1
1183     2765 1
1184     2766 1
1185     2767 1
1186     2768 1
1187     2769 1
1188     2770 1
1189     2771 1
1190     2772 1
1191     2773 1
1192     2774 1
1193     2775 1
1194     2776 1
1195     2777 1
1196     2778 1
1197     2779 1
1198     2780 1
1199     2781 1
1200     2782 1
1201     2783 1
1202     2784 1
1203     2785 1
1204     2786 1
1205     2787 1
1206     2788 1
1207     2789 1
1208     2790 1
1209     2791 1
1210     2792 1
1211     2793 1
1212     2794 1
1213     2795 1
1214     2796 1
1215     2797 1
1216     2798 1
1217     2799 1
1218     2800 1
1219     2801 1
1220     2802 1
1221     2803 1
1222     2804 1
1223     2805 1
1224     2806 1
1225     2807 1
1226     2808 1
1227     2809 1
1228     2810 1
1229     2811 1
1230     2812 1
1231     2813 1
1232     2814 1
1233     2815 1
1234     2816 1
1235     2817 1
1236     2818 1
1237     2819 1
1238     2820 1
1239     2821 1
1240     2822 1
1241     2823 1
1242     2824 1
1243     2825 1
1244     2826 1
1245     2827 1
1246     2828 1
1247     2829 1
1248     2830 1
1249     2831 1
1250     2832 1
1251     2833 1
1252     2834 1
1253     2835 1
1254     2836 1
1255     2837 1
1256     2838 1
1257     2839 1
1258     2840 1
1259     2841 1
1260     2842 1
1261     2843 1
1262     2844 1
1263     2845 1
1264     2846 1
1265     2847 1
1266     2848 1
1267     2849 1
1268     2850 1
1269     2851 1
1270     2852 1
1271     2853 1
1272     2854 1
1273     2855 1
1274     2856 1
1275     2857 1
1276     2858 1
1277     2859 1
1278     2860 1
1279     2861 1
1280     2862 1
1281     2863 1
1282     2864 1
1283     2865 1
1284     2866 1
1285     2867 1
1286     2868 1
1287     2869 1
1288     2870
```

```

763      2344 2 BEGIN
764      2345 2
765      2346 2 BIND
766      2347 2
767      2348 2      PREFIX_LEN      = .P_PREFIX_LEN,      ! holds length of prefix buffer
768      2349 2      SUFFIX_LEN      = .P_SUFFIX_LEN;      ! holds length of suffix buffer
769      2350 2
770      2351 2 LOCAL
771      2352 2
772      2353 2      BUFFER_PTR;
773      2354 2
774      2355 2 MACRO
775      2356 2
776      2357 2      VT100_OFF = %STRING (%CHAR (ESC), %CHAR (LB), '0m')%;
777      2358 2
778      2359 2      BUFFER_PTR = .PREFIX_BUF + .PREFIX_LEN; ! init to first free byte of prefix
779      2360 2
780      2361 2 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
781      2362 2      SET
782      2363 2      [HARDCOPY, UNKNOWN, VT05, VT52, VTFOREIGN];
783      2364 2      BEGIN
784      2365 2      !+
785      2366 2      ! Renditions not supported on these devices. Just return.
786      2367 2
787      2368 2      RETURN      SSS_NORMAL
788      2369 2      END;
789      2370 2
790      2371 2      [INRANGE, OUTRANGE];
791      2372 2      RETURN 0;          ! error
792      2373 2
793      2374 2
794      2375 2
795      2376 2      IF .FLAGS <0,4> EQ 0
796      2377 2      THEN
797      2378 2      RETURN (SSS_NORMAL); ! no attributes requested
798      2379 2
799      2380 2
800      2381 2      !+ For each attribute bit set in flags, copy
801      2382 2      ! the appropriate ASCII graphic rendition byte
802      2383 2      ! followed by a ';' into the output buffer.
803      2384 2      ! Note use of autoincrementing.
804      2385 2
805      2386 2
806      2387 2      CHSWCHAR_A (ESC, BUFFER_PTR);
807      2388 2      CHSWCHAR_A (LB, BUFFER_PTR);
808      2389 2      PREFIX_LEN = .PREFIX_LEN + 2; ! Start with 2 chars: <ESC> "["
809      2390 2      INCR I-FROM 0 TO 3
810      2391 2      DO
811      2392 2      BEGIN          ! build prefix attribute string
812      2393 2      BIND
813      2394 2      ATTRTABL = UPLIT (BYTE ('1754')) : VECTOR [4, BYTE];
814      2395 2
815      2396 2      IF .FLAGS <.1, 1>
816      2397 2      THEN
817      2398 2      BEGIN
818      2399 2      CHSWCHAR_A (.ATTRTABL[I], BUFFER_PTR);
819      2400 2

```

```

820 2401 5
821 2402 5
822 2403 5
823 2404 5
824 2405 5
825 2406 5
826 2407 5
827 2408 5
828 2409 5
829 2410 5
830 2411 5
831 2412 5
832 2413 5
833 2414 5
834 2415 5
835 2416 5
836 2417 5
837 2418 5
838 2419 5
839 2420 5
840 2421 5
841 2422 5
842 2423 5
843 2424 5
844 2425 5
845 2426 5
846 2427 5
847 2428 5
848 2429 5
849 2430 5
850 2431 5
851 2432 5
852 2433 5

    PREFIX_LEN = .PREFIX_LEN + 2; ! keep updating length
    END;
    ! build prefix attribute string

    !+ When we fall out of above loop we have deposited
    ! an extra ';' at the end of the buffer. Back up
    ! FREE_ADDR and write VT100_SGR on top of it.

    BUFFER_PTR = .BUFFER_PTR - 1;
    CHSWCHAR_A (VT100_SGR, BUFFER_PTR);

    END;
    TES;

    !+ Append in sequence to turn off graphic rendition.

    !- BUFFER_PTR = .SUFFIX_BUF + .SUFFIX_LEN; ! init to first free byte in
    ! suffix buffer.

    CHSMOVE (%CHARCOUNT (VT100_OFF), UPLIT (BYTE (VT100_OFF)), .BUFFER_PTR);

    !+ Set the output length and exit.
    !- SUFFIX_LEN = .SUFFIX_LEN + %CHARCOUNT(VT100_OFF);

    RETURN SSS_NORMAL

    ! End of routine COB$SET_ATTRIBUTES_ONLY

```

34	35	37	31	001DA	P.AAR:	.BLKB	2
6D	30	5B	1B	001DC	P.AAS:	.ASCII	\1754\
				001E0		<27>\[0m\]	

ATTRTABL= P.AAR

000E	0046	51	52	10	AC	0004	00000	.ENTRY	COB\$SET_ATTRIBUTES_ONLY, Save R2	2293
		05	AC		62	C1	00006	MOVL	P PREFIX-LEN, R2	2348
		00	00	04	AC	CF	0000B	ADDL3	(R2), PREFIX BUF BUFFER_PTR	2359
		0046	0046		0046		00010	CASEL	TERM TYPE, #0, #5	2361
		0046	0046		0046		00018	.WORD	\$S-1\$,-	
									\$S-1\$,-	
									\$S-1\$,-	
									2\$-1\$,-	
									\$S-1\$,-	
									\$S-1\$,-	
									\$S-1\$	
								BRB	6\$	2372
		0F	08	3C	93	11	0001C	BITB	FLAGS, #15	2376
					32	13	00022	BEQL	\$S	
		81	581B	8F	80	00024	MOVW	#23323, (BUFFER_PTR)+		2387

		62	02	C0 00029	ADDL2	#2, (R2)	: 2389
			50	D4 0002C	CLRL	I	: 2390
08	08	AC	50	E1 0002E	BBC	FLAGS, 48	: 2396
		81	C1 AF40	90 00033	MOVB	ATTRTABL[I], (BUFFER_PTR)+	: 2399
		81	38	90 00038	MOVB	#59, (BUFFER_PTR)+	: 2400
		EC	62	02	ADDL2	#2, (R2)	: 2401
		50	03	C0 0003B	AOBLEQ	#3, I, 38	: 2390
		71	6D	F3 0003E	MOVB	#109, -(BUFFER_PTR)	: 2411
51	14	AC	51	D6 00040	INCL	BUFFER_PTR	: 2420
		61	BC	C1 00048	ADDL3	#P_SUFFIX_LEN, SUFFIX_BUF, BUFFER_PTR	: 2423
		18	AB	AF D0 0004E	MOVL	P_XAS, (BUFFER_PTR)	: 2429
		50	04	C0 00052	ADDL2	#4, #P_SUFFIXLEN	: 2431
			01	D0 00056	MOVL	#1, R0	: 2433
				04 00059	RET	RET	
				50 D4 0005A	CLRL	RO	
				04 0005C	RET		

; Routine Size: 93 bytes. Routine Base: \_COB\$CODE + 01E4

; 853 2434 1 !<BLF/PAGE>

```

855 2435 1 %SBTTL 'COB$SET_CURSOR_ABS_R4 - Create absolute set cursor sequence'
856 2436 1 GLOBAL ROUTINE COB$SET_CURSOR_ABS_R4 (
857 2437 1      TERM_TYPE,
858 2438 1      LINE_NO,
859 2439 1      COL_NO,
860 2440 1      BUFFER,
861 2441 1      CUR_SIZE
862 2442 1      ) : COB$SET_R4_LNK =
863 2443 1
864 2444 1 ** FUNCTIONAL DESCRIPTION:
865 2445 1
866 2446 1 This routine generates the escape sequence for a set cursor
867 2447 1 position and appends the string to a given output buffer.
868 2448 1
869 2449 1 ** CALLING SEQUENCE:
870 2450 1
871 2451 1     ret_status.wlc.v = COB$SET_CURSOR_ABS_R4 (TERM_TYPE.rl.v, LINE_NO.rl.v,
872 2452 1             COL_NO.rl.v, BUFFER.mt.r,
873 2453 1             CUR_SIZE.ml.r)
874 2454 1
875 2455 1 ** FORMAL PARAMETERS:
876 2456 1
877 2457 1     TERM_TYPE.rl.v      terminal type
878 2458 1     LINE_NO.rl.v      line number
879 2459 1     COL_NO.rl.v      column number
880 2460 1     BUFFER.mt.r      addr of buffer
881 2461 1             this buffer should be at least
882 2462 1             20 bytes
883 2463 1     CUR_SIZE.ml.r      # bytes currently in buffer
884 2464 1
885 2465 1 ** IMPLICIT INPUTS:
886 2466 1     NONE
887 2467 1
888 2468 1 ** IMPLICIT OUTPUTS:
889 2469 1
890 2470 1
891 2471 1     NONE
892 2472 1
893 2473 1 ** COMPLETION STATUS:
894 2474 1
895 2475 1
896 2476 1 ** SIDE EFFECTS:
897 2477 1     NONE
898 2478 1
899 2479 1 -- BEGIN
900 2480 1
901 2481 2 LOCAL
902 2482 2
903 2483 2
904 2484 2     VT100CTL : VECTOR [1, 8] INITIAL (
905 2485 2             DSC$K_CLASS S ^24 + DSC$K_DTYPE T ^16 + 10,
906 2486 2             UPLIT( BYTE (ESC, LB, '!OL;!UL', VT100_SC ))),
907 2487 2             ; dsc for cvt to vt100 sequence
908 2488 2             ; FAO control string
909 2489 2     FREE_ADDR : REF VECTOR [,BYTE]; ; addr of 1st free byte
910 2490 2
911 2491 2

```

COB\$SESCAPE\_GEN COB\$SESCAPE\_GENERATOR - Escape sequence generat 16-Sep-1984 00:06:34  
 1-003 COB\$SET\_CURSOR\_ABS\_R4 - Create absolute set cu 14-Sep-1984 12:10:44 L 12  
 VAX-11 Bliss-32 v6.0-742  
 [COBRTL.SRC]COBESCGEN.B32:1

```

912      2492 2      FREE_ADDR = .BUFFER + ..CUR_SIZE; ! addr of next free byte
913      2493 2
914      2494 2
915      2495 2
916      2496 2
917      2497 2
918      2498 2
919      2499 2
920      2500 2
921      2501 2
922      2502 2
923      2503 2
924      2504 2
925      2505 2
926      2506 2
927      2507 2
928      2508 2
929      2509 2
930      2510 2
931      2511 2
932      2512 2
933      2513 2
934      2514 2
935      2515 2
936      2516 2
937      2517 2
938      2518 2
939      2519 2
940      2520 2
941      2521 2
942      2522 2
943      2523 2
944      2524 2
945      2525 2
946      2526 2
947      2527 2
948      2528 2
949      2529 2
950      2530 2
951      2531 2
952      2532 2
953      2533 2
954      2534 2
955      2535 2
956      2536 2
957      2537 2
958      2538 2
959      2539 2
960      2540 2
961      2541 2
962      2542 2
963      2543 2
964      2544 2
965      2545 2
966      2546 2
967      2547 2
968      2548 2
  
```

FREE\_ADDR = .BUFFER + ..CUR\_SIZE; ! addr of next free byte  
 CASE .TERM\_TYPE FROM UNKNOWN TO HARDCOPY OF  
 SET  
 [HARDCOPY, UNKNOWN, VTFOREIGN]:  
 ; ! do nothing  
 [VT05]:  
 BEGIN  
 .CUR\_SIZE = ..CUR\_SIZE + 3; ! update current size of buffer  
 FREE\_ADDR [0] = VT05\_SC; ! put set cursor sequence into buffer  
 FREE\_ADDR [1] = CB + .LINE\_NO;  
 FREE\_ADDR [2] = CB + .COL\_NO;  
 END;  
 [VT52]:  
 BEGIN  
 .CUR\_SIZE = ..CUR\_SIZE + 4; ! update current size of buffer  
 FREE\_ADDR [0] = ESC; ! put set cursor sequence into buffer  
 FREE\_ADDR [1] = VT52\_SC;  
 FREE\_ADDR [2] = CB + .LINE\_NO;  
 FREE\_ADDR [3] = CB + .COL\_NO;  
 END;  
 [VT100]:  
 BEGIN  
 LOCAL  
 STATUS,  
 CVT\_ARGS : VECTOR [2],  
 FAO\_BUFFER : BLOCK [8, BYTE],  
 FAO\_LEN : WORD;  
 CVT\_ARGS [0] = .LINE\_NO;  
 CVT\_ARGS [1] = .COL\_NO;  
 FAO\_BUFFER [DSC\$B\_DTYPE] = DSC\$K\_DTYPE\_T;  
 FAO\_BUFFER [DSC\$B\_CLASS] = DSC\$K\_CLASS\_S;  
 FAO\_BUFFER [DSC\$W\_LENGTH] = 20; ! arbitrary - sb large enough  
 FAO\_BUFFER [DSC\$A\_POINTER] = .FREE\_ADDR;  
 !+  
 ! Convert to ASCII characters and move to buffer.  
 STATUS = SFAOL (CTRSTR = VT100CTL, OUTLEN = FAO\_LEN,  
 OUTBUF = FAO\_BUFFER, PRMLST = CVT\_ARGS);  
 IF NOT .STATUS THEN RETURN (.STATUS);  
 .CUR\_SIZE = ..CUR\_SIZE + .FAO\_LEN; ! add length of appended string  
 END;  
 [INRANGE, OUTRANGE]:  
 RETURN 0; ! should never get here  
 TES;  
 RETURN 1;

: 969 2549 1 END:

! End of routine COBSSSET\_CURSOR\_ABS\_R4

: Routine Size: 130 bytes, Routine Base: \_COBSCODE + 024E

: 970 2550 1 !<BLF/PAGE>

```

    972      2551 1 ZSBTTL 'COB$SET_CURSOR_REL Create relative cursor position sequence'
    973      2552 1 GLOBAL ROUTINE COB$SET_CURSOR_REL (
    974          2553 1 TERM_TYPE,
    975          2554 1 LINE_NO,
    976          2555 1 COL_NO,
    977          2556 1 LINE_PLUS,
    978          2557 1 COL_PLUS,
    979          2558 1 BUFFER,
    980          2559 1 CUR_SIZE
    981          2560 1 ) =
    982          2561 1 /**
    983          2562 1 FUNCTIONAL DESCRIPTION:
    984
    985          2563 1
    986          2564 1 This routine generates the escape sequence to position
    987          2565 1 the cursor relative to the specified line and column, or
    988          2566 1 relative to the current position if none is specified.
    989          2567 1 The set cursor sequence is appended to the output string.
    990          2568 1
    991          2569 1 Notice that the ANSI sequences can become quite large.
    992          2570 1 For instance, it is possible that 50 up arrows (2 bytes each)
    993          2571 1 will be only a part of the resulting sequence. It is
    994          2572 1 recommended that the output buffer be 512 bytes long.
    995          2573 1
    996          2574 1 CALLING SEQUENCE:
    997          2575 1
    998          2576 1     ret_status.wlc.v = COB$SET_CURSOR_REL (TERM_TYPE.rl.v, LINE_NO.rl.v,
    999          2577 1                               COL_NO.rl.v, LINE_PLUS.rl.v,
    1000         2578 1                               COL_PLUS.rl.v, BUFFER.mt.r,
    1001         2579 1                               CUR_SIZE.ml.r)
    1002         2580 1
    1003         2581 1 FORMAL PARAMETERS:
    1004         2582 1
    1005         2583 1     TERM_TYPE.rl.v          terminal type
    1006         2584 1     LINE_NO.rl.v          line number
    1007         2585 1     COL_NO.rl.v          column number
    1008         2586 1     LINE_PLUS.rl.v        offset from line number
    1009         2587 1     COL_PLUS.rl.v        offset from column number
    1010         2588 1     BUFFER.mt.r          addr of buffer
    1011         2589 1     CUR_SIZE.ml.r        # bytes currently in buffer
    1012         2590 1
    1013         2591 1 IMPLICIT INPUTS:
    1014         2592 1     NONE
    1015         2593 1
    1016         2594 1 IMPLICIT OUTPUTS:
    1017         2595 1     NONE
    1018         2596 1
    1019         2597 1
    1020         2598 1
    1021         2599 1
    1022         2600 1
    1023         2601 1
    1024         2602 1
    1025         2603 1
    1026         2604 1
    1027         2605 1
    1028         2606 1
    1029         2607 1
    1030
    1031
    1032
    1033
    1034
    1035
    1036
    1037
    1038
    1039
    1040
    1041
    1042
    1043
    1044
    1045
    1046
    1047
    1048
    1049
    1050
    1051
    1052
    1053
    1054
    1055
    1056
    1057
    1058
    1059
    1060
    1061
    1062
    1063
    1064
    1065
    1066
    1067
    1068
    1069
    1070
    1071
    1072
    1073
    1074
    1075
    1076
    1077
    1078
    1079
    1080
    1081
    1082
    1083
    1084
    1085
    1086
    1087
    1088
    1089
    1090
    1091
    1092
    1093
    1094
    1095
    1096
    1097
    1098
    1099
    1100
    1101
    1102
    1103
    1104
    1105
    1106
    1107
    1108
    1109
    1110
    1111
    1112
    1113
    1114
    1115
    1116
    1117
    1118
    1119
    1120
    1121
    1122
    1123
    1124
    1125
    1126
    1127
    1128
    1129
    1130
    1131
    1132
    1133
    1134
    1135
    1136
    1137
    1138
    1139
    1140
    1141
    1142
    1143
    1144
    1145
    1146
    1147
    1148
    1149
    1150
    1151
    1152
    1153
    1154
    1155
    1156
    1157
    1158
    1159
    1160
    1161
    1162
    1163
    1164
    1165
    1166
    1167
    1168
    1169
    1170
    1171
    1172
    1173
    1174
    1175
    1176
    1177
    1178
    1179
    1180
    1181
    1182
    1183
    1184
    1185
    1186
    1187
    1188
    1189
    1190
    1191
    1192
    1193
    1194
    1195
    1196
    1197
    1198
    1199
    1200
    1201
    1202
    1203
    1204
    1205
    1206
    1207
    1208
    1209
    1210
    1211
    1212
    1213
    1214
    1215
    1216
    1217
    1218
    1219
    1220
    1221
    1222
    1223
    1224
    1225
    1226
    1227
    1228
    1229
    1230
    1231
    1232
    1233
    1234
    1235
    1236
    1237
    1238
    1239
    1240
    1241
    1242
    1243
    1244
    1245
    1246
    1247
    1248
    1249
    1250
    1251
    1252
    1253
    1254
    1255
    1256
    1257
    1258
    1259
    1260
    1261
    1262
    1263
    1264
    1265
    1266
    1267
    1268
    1269
    1270
    1271
    1272
    1273
    1274
    1275
    1276
    1277
    1278
    1279
    1280
    1281
    1282
    1283
    1284
    1285
    1286
    1287
    1288
    1289
    1290
    1291
    1292
    1293
    1294
    1295
    1296
    1297
    1298
    1299
    1300
    1301
    1302
    1303
    1304
    1305
    1306
    1307
    1308
    1309
    1310
    1311
    1312
    1313
    1314
    1315
    1316
    1317
    1318
    1319
    1320
    1321
    1322
    1323
    1324
    1325
    1326
    1327
    1328
    1329
    1330
    1331
    1332
    1333
    1334
    1335
    1336
    1337
    1338
    1339
    1340
    1341
    1342
    1343
    1344
    1345
    1346
    1347
    1348
    1349
    1350
    1351
    1352
    1353
    1354
    1355
    1356
    1357
    1358
    1359
    1360
    1361
    1362
    1363
    1364
    1365
    1366
    1367
    1368
    1369
    1370
    1371
    1372
    1373
    1374
    1375
    1376
    1377
    1378
    1379
    1380
    1381
    1382
    1383
    1384
    1385
    1386
    1387
    1388
    1389
    1390
    1391
    1392
    1393
    1394
    1395
    1396
    1397
    1398
    1399
    1400
    1401
    1402
    1403
    1404
    1405
    1406
    1407
    1408
    1409
    1410
    1411
    1412
    1413
    1414
    1415
    1416
    1417
    1418
    1419
    1420
    1421
    1422
    1423
    1424
    1425
    1426
    1427
    1428
    1429
    1430
    1431
    1432
    1433
    1434
    1435
    1436
    1437
    1438
    1439
    1440
    1441
    1442
    1443
    1444
    1445
    1446
    1447
    1448
    1449
    1450
    1451
    1452
    1453
    1454
    1455
    1456
    1457
    1458
    1459
    1460
    1461
    1462
    1463
    1464
    1465
    1466
    1467
    1468
    1469
    1470
    1471
    1472
    1473
    1474
    1475
    1476
    1477
    1478
    1479
    1480
    1481
    1482
    1483
    1484
    1485
    1486
    1487
    1488
    1489
    1490
    1491
    1492
    1493
    1494
    1495
    1496
    1497
    1498
    1499
    1500
    1501
    1502
    1503
    1504
    1505
    1506
    1507
    1508
    1509
    1510
    1511
    1512
    1513
    1514
    1515
    1516
    1517
    1518
    1519
    1520
    1521
    1522
    1523
    1524
    1525
    1526
    1527
    1528
    1529
    1530
    1531
    1532
    1533
    1534
    1535
    1536
    1537
    1538
    1539
    1540
    1541
    1542
    1543
    1544
    1545
    1546
    1547
    1548
    1549
    1550
    1551
    1552
    1553
    1554
    1555
    1556
    1557
    1558
    1559
    1560
    1561
    1562
    1563
    1564
    1565
    1566
    1567
    1568
    1569
    1570
    1571
    1572
    1573
    1574
    1575
    1576
    1577
    1578
    1579
    1580
    1581
    1582
    1583
    1584
    1585
    1586
    1587
    1588
    1589
    1590
    1591
    1592
    1593
    1594
    1595
    1596
    1597
    1598
    1599
    1600
    1601
    1602
    1603
    1604
    1605
    1606
    1607
    1608
    1609
    1610
    1611
    1612
    1613
    1614
    1615
    1616
    1617
    1618
    1619
    1620
    1621
    1622
    1623
    1624
    1625
    1626
    1627
    1628
    1629
    1630
    1631
    1632
    1633
    1634
    1635
    1636
    1637
    1638
    1639
    1640
    1641
    1642
    1643
    1644
    1645
    1646
    1647
    1648
    1649
    1650
    1651
    1652
    1653
    1654
    1655
    1656
    1657
    1658
    1659
    1660
    1661
    1662
    1663
    1664
    1665
    1666
    1667
    1668
    1669
    1670
    1671
    1672
    1673
    1674
    1675
    1676
    1677
    1678
    1679
    1680
    1681
    1682
    1683
    1684
    1685
    1686
    1687
    1688
    1689
    1690
    1691
    1692
    1693
    1694
    1695
    1696
    1697
    1698
    1699
    1700
    1701
    1702
    1703
    1704
    1705
    1706
    1707
    1708
    1709
    1710
    1711
    1712
    1713
    1714
    1715
    1716
    1717
    1718
    1719
    1720
    1721
    1722
    1723
    1724
    1725
    1726
    1727
    1728
    1729
    1730
    1731
    1732
    1733
    1734
    1735
    1736
    1737
    1738
    1739
    1740
    1741
    1742
    1743
    1744
    1745
    1746
    1747
    1748
    1749
    1750
    1751
    1752
    1753
    1754
    1755
    1756
    1757
    1758
    1759
    1760
    1761
    1762
    1763
    1764
    1765
    1766
    1767
    1768
    1769
    1770
    1771
    1772
    1773
    1774
    1775
    1776
    1777
    1778
    1779
    1780
    1781
    1782
    1783
    1784
    1785
    1786
    1787
    1788
    1789
    1790
    1791
    1792
    1793
    1794
    1795
    1796
    1797
    1798
    1799
    1800
    1801
    1802
    1803
    1804
    1805
    1806
    1807
    1808
    1809
    1810
    1811
    1812
    1813
    1814
    1815
    1816
    1817
    1818
    1819
    1820
    1821
    1822
    1823
    1824
    1825
    1826
    1827
    1828
    1829
    1830
    1831
    1832
    1833
    1834
    1835
    1836
    1837
    1838
    1839
    1840
    1841
    1842
    1843
    1844
    1845
    1846
    1847
    1848
    1849
    1850
    1851
    1852
    1853
    1854
    1855
    1856
    1857
    1858
    1859
    1860
    1861
    1862
    1863
    1864
    1865
    1866
    1867
    1868
    1869
    1870
    1871
    1872
    1873
    1874
    1875
    1876
    1877
    1878
    1879
    1880
    1881
    1882
    1883
    1884
    1885
    1886
    1887
    1888
    1889
    1890
    1891
    1892
    1893
    1894
    1895
    1896
    1897
    1898
    1899
    1900
    1901
    1902
    1903
    1904
    1905
    1906
    1907
    1908
    1909
    1910
    1911
    1912
    1913
    1914
    1915
    1916
    1917
    1918
    1919
    1920
    1921
    1922
    1923
    1924
    1925
    1926
    1927
    1928
    1929
    1930
    1931
    1932
    1933
    1934
    1935
    1936
    1937
    1938
    1939
    1940
    1941
    1942
    1943
    1944
    1945
    1946
    1947

```

1029	2608	1	combination of the LINE and COLUMN phrases on both ANSI devices			
1030	2609	1	and VT100s. The arrows on the VT52 can only be moved one position at			
1031	2610	1	a time. This may be slower, but at least the results will be the			
1032	2611	1	same as far as cursor positioning goes on both types of terminals.			
1033	2612	1				
1034	2613	1	'v' = down arrow			
1035	2614	1				
1036	2615	1	'^' = up arrow			
1037	2616	1				
1038	2617	1	LINE a : LINE PLUS b : COLUMN c : COLUMN PLUS d : Cursor Pos. Used			
1039	2618	1	-----			
1040	2619	1				
1041	2620	1	N	N	N	Current Rules
1042	2621	1	N	NN	NN	d "->"
1043	2622	1	N	NN	NN	<CR> : c-1 "->"
1044	2623	1	N	YY	YY	<CR> : (c-1)+d "->"
1045	2624	1	NN	YY	NN	b <LF>
1046	2625	1	NN	YY	NN	b <LF> : d "->"
1047	2626	1	NN	YY	NN	b <LF> : <CR> : c-1 "->"
1048	2627	1	N	YY	YY	b <LF> : <CR> : (c-1)+d "->"
1049	2628	1	YY	NN	NN	Home ; a-1 "v"
1050	2629	1	YY	NN	NN	24 "a"; a-1 "v" ; d "->"
1051	2630	1	YY	NN	NN	Direct a,c
1052	2631	1	YY	NY	YY	Direct a,c+d
1053	2632	1	YY	YY	NN	Home ; a-1 "v"; b "LF"
1054	2633	1	Y	Y	NN	24 "a"; a-1 "v"; b <LF>
1055	2634	1			Y	d "->"
1056	2635	1	Y	Y	Y	Direct a,c ; b <LF>
1057	2636	1	Y	Y	Y	Direct a,c+d ; b <LF>
1058	2637	1				-----
1059	2638	1				
1060	2639	1				
1061	2640	1	note: <lf> for all LINE PLUS to get scrolling			
1062	2641	1	note: 24 up arrows used instead of home - this maintains the current			
1063	2642	1	column position			
1064	2643	1	-----			

```

1066      2644 1
1067      2645 2
1068      2646 2
1069      2647 2
1070      2648 2
1071      2649 2
1072      2650 2
1073      2651 2
1074      2652 2
1075      2653 2
1076      2654 2
1077      2655 2
1078      2656 2
1079      2657 2
1080      2658 2
1081      2659 2
1082      2660 2
1083      2661 2
1084      2662 2
1085      2663 2
1086      2664 2
1087      2665 2
1088      2666 2
1089      2667 2
1090      2668 2
1091      2669 2
1092      2670 2
1093      2671 2
1094      2672 2
1095      2673 2
1096      2674 2
1097      2675 2
1098      2676 2
1099      2677 2
1100      2678 2
1101      2679 2
1102      2680 2
1103      2681 2
1104      2682 2
1105      2683 2
1106      2684 2
1107      2685 2
1108      2686 2
1109      2687 2
1110      2688 2
1111      2689 2
1112      2690 2
1113      2691 2
1114      2692 2
1115      2693 2
1116      2694 2
1117      2695 2
1118      2696 2
1119      2697 2
1120      2698 2
1121      2699 2
1122      2700 2

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122

BEGIN
!+
| The following macro will put the VT100 sequence for
| multiple arrow movement into the buffer and update
| the length and pointer. Sequences are of the form
| ESC [ num arrow.
!-
MACRO
  SAPPEND_VT100_SEQ (NUM, CTR_ARROW) =
  BEGIN
  LOCAL
    CVT_ARG,
    FAO_BUF : BLOCK [8, BYTE],
    FAO_LEN : WORD,
    STATUS;
  IF NUM NEQ 0
  THEN
    BEGIN
    CVT_ARG = NUM;
    FAO_BUF [DSC$B_DTYPE] = DSC$K_DTYPE_T;
    FAO_BUF [DSC$B_CLASS] = DSC$K_CLASS_S;
    FAO_BUF [DSC$W_LENGTH] = 15;           ! arbitrary - sb big enough
    FAO_BUF [DSC$A_POINTER] = .FREE_ADDR;
    STATUS = $FAOL (CTRSTR = CTR_ARROW, OUTLEN = FAO_LEN,
                    OUTBUF = FAO_BUF, PRMLST = CVT_ARG);
    IF NOT .STATUS THEN RETURN .STATUS;
    .CUR_SIZE = ..CUR_SIZE + .FAO_LEN;
    FREE_ADDR = .FREE_ADDR + .FAO_LEN;
    END;
  END
  ! end macro Sappend_vt100_seq

!+
| This macro puts NUM arrows into the buffer.
| The next free byte and buffer size are updated.
!-
MACRO
  SAPPEND_N_ARROWS (NUM, DIRECTION) =
  BEGIN
  INCR COUNTER FROM 1 TO NUM DO
    BEGIN
    FREE_ADDR = CHSMOVE (2, UPLIT (BYTE (ESC, DIRECTION)), .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 2;
    END;
  END;
  ! end of macro append_n_arrows

MACRO
  SAPPEND_VT100_HOME =
  BEGIN
  FREE_ADDR = CHSMOVE (3, UPLIT (BYTE (ESC, LB, f)),
                        .FREE_ADDR);

```

```
1123      M 2701 2      .CUR_SIZE = ..CUR_SIZE + 3;
1124      M 2702 2      END
1125      M 2703 2
1126      M 2704 2
1127      M 2705 2
1128      M 2706 2      MACRO
1129      M 2707 2      SAPPEND_VT52_HOME =
1130      M 2708 2      BEGIN
1131      M 2709 2      FREE_ADDR = CHSMOVE (2, UPLIT (BYTE (ESC, H)), .FREE_ADDR);
1132      M 2710 2      .CUR_SIZE = ..CUR_SIZE + 2;
1133      M 2711 2      END;
1134      M 2712 2
1135      M 2713 2
1136      M 2714 2      LOCAL
1137      M 2715 2      FREE_ADDR : REF VECTOR [,BYTE];
1138      M 2716 2      UP_CTL : VECTOR [1, 8] INITIAL (
1139      M 2717 2      DSCSK_CLASS_S ^ 24 + DSCSK_DTYPE_T ^ 16 + 6,
1140      M 2718 2      UPLIT (BYTE (ESC, LB, '!UL', A));
1141      M 2719 2      DOWN_CTL : VECTOR [1, 8] INITIAL (
1142      M 2720 2      DSCSK_CLASS_S ^ 24 + DSCSK_DTYPE_T ^ 16 + 6,
1143      M 2721 2      UPLIT (BYTE (ESC, LB, '!UL', B));
1144      M 2722 2      RIGHT_CTL : VECTOR [1, 8] INITIAL (
1145      M 2723 2      DSCSK_CLASS_S ^ 24 + DSCSK_DTYPE_T ^ 16 + 6,
1146      M 2724 2      UPLIT (BYTE (ESC, LB, '!UL', C));
1147      M 2725 2      BIND
1148      M 2726 2      UP = A,
1149      M 2727 2      DOWN = B,
1150      M 2728 2      RIGHT = C;
1151      M 2729 2
1152      M 2730 2
1153      M 2731 2      LITERAL
1153      M 2731 2      K_MAX_RMS_SIZE = 255;
```

! equate letters to directions

```

1155      2732 2 IF .TERM_TYPE NEQ VT100 AND
1156      2733 2   .TERM_TYPE NEQ VT52
1157      2734 2 THEN RETURN (SSS_NORMAL);      ! don't do anything for other
1158      2735 2                                     ! terminal types
1159      2736 2
1160      2737 2 FREE_ADDR = .BUFFER + ..CUR_SIZE;
1161      2738 2
1162      2739 2 IF .LINE_NO NEQ 0 AND
1163      2740 2   .COL_NO NEQ 0
1164      2741 2 THEN                                     ! direct cursor addressing
1165      2742 2   BEGIN
1166      2743 2     COB$SET_CURSOR_ABS_R4 (.TERM_TYPE, .LINE_NO,
1167      2744 2           .COL_NO + .COL_PLUS, .BUFFER,
1168      2745 2           .CUR_SIZE);
1169      2746 2   FREE_ADDR = .BUFFER + ..CUR_SIZE; ! update addr next free byte
1170      2747 2 END;
1171      2748 2
1172      2749 2 IF .LINE_NO NEQ 0 AND
1173      2750 2   .COL_NO EQL 0
1174      2751 2 THEN
1175      2752 2   BEGIN
1176      2753 2     IF .COL_PLUS EQL 0           ! insert home sequence
1177      2754 2     THEN
1178      2755 2       BEGIN
1179      2756 2         IF .TERM_TYPE EQL VT100
1180      2757 2         THEN
1181      2758 2           SAPPEND_VT100_HOME
1182      2759 2         ELSE
1183      2760 2           SAPPEND_VT52_HOME;
1184      2761 2       END
1185      2762 2     ELSE
1186      2763 2       BEGIN           ! insert a bunch of up arrows
1187      2764 2         MACRO
1188      2765 2         UP_ARROW = %STRING (%CHAR (ESC), %CHAR (A));
1189      2766 2         BIND
1190      2767 2         UP_24 = UPLIT (BYTE (REP 24 OF (UP_ARROW)));
1191      2768 2
1192      2769 2     IF .TERM_TYPE EQL VT100
1193      2770 2     THEN
1194      2771 2       SAPPEND_VT100_SEQ (24, UP_CTL)
1195      2772 2     ELSE
1196      2773 2       BEGIN
1197      2774 2         FREE_ADDR = CH$MOVE (48, UP_24, .FREE_ADDR);
1198      2775 2         .CUR_SIZE = ..CUR_SIZE + 48;
1199      2776 2       END;
1200      2777 2
1201      2778 2
1202      2779 2   !+ Insert line_no down arrows regardless of col_plus
1203      2780 2
1204      2781 2   IF .TERM_TYPE EQL VT100
1205      2782 2   THEN
1206      2783 2     SAPPEND_VT100_SEQ (.LINE_NO - 1, DOWN_CTL)
1207      2784 2   ELSE
1208      2785 2     SAPPEND_N_ARROWS (.LINE_NO - 1, DOWN);
1209      2786 2
1210      2787 2
1211      2788 2 IF .LINE_NO EQL 0 AND

```

```

1212 2789 2 .COL_NO NEQ 0
1213 2790 2 THEN BEGIN ! insert a CR &
1214 2791 2 .COL_NO right arrows
1215 2792 2 FREE_ADDR [0] = CR;
1216 2793 2 FREE_ADDR = .FREE_ADDR + 1;
1217 2794 2 .CUR_SIZE = ..CUR_SIZE + 1;
1218 2795 2 END;
1219 2796 2
1220 2797 2 IF .LINE_PLUS NEQ 0 ! add line_plus LFs to buffer
1221 2798 2 THEN BEGIN
1222 2799 2 FREE_ADDR = CHSFILL (LF, .LINE_PLUS, .FREE_ADDR);
1223 2800 2 .CUR_SIZE = ..CUR_SIZE + .LINE_PLUS;
1224 2801 2 END;
1225 2802 2
1226 2803 2
1227 2804 2 IF (.COL_PLUS NEQ 0 OR .COL_NO NEQ 0) AND ! didn't do direct cursor addr
1228 2805 2 (.LINE_NO EQL 0 OR .COL_NO EQL 0) ! insert col_plus right arrows
1229 2806 2 THEN BEGIN
1230 2807 2 LOCAL
1231 2808 2 COL;
1232 2809 2 COL = .COL_NO - 1;
1233 2810 2 IF .COL LSS 0
1234 2811 2 THEN
1235 2812 2 COL = 0;
1236 2813 2 IF .TERM_TYPE EQL VT100
1237 2814 2 THEN $APPEND_VT100_SEQ (.COL + .COL_PLUS, RIGHT_CTL)
1238 2815 2 ELSE $APPEND_N_ARROWS (.COL + .COL_PLUS, RIGHT);
1239 2816 2
1240 2817 2
1241 2818 2
1242 2819 2
1243 2820 2
1244 2821 2 RETURN (SSS_NORMAL); ! everything should be in the buffer
1245 2822 2
1246 2823 1 END; ! End of routine COB$SET_CURSOR_REL

```

4C	5B	1B	002D0	P.AAU:	.BYTE	27, 91
			002D2	.ASCII	\\UL\\	
			41	.BYTE	65	
			002D5	.BYTE	66	
			002D6	.BLKB	2	
4C	5B	1B	002D8	P.AAV:	.BYTE	27, 91
			002DA	.ASCII	\\UL\\	
			42	.BYTE	66	
			002DD	.BYTE	67	
			002DE	.BLKB	2	
4C	55	21	002E0	P.AAW:	.BYTE	27, 91
			002E2	.ASCII	\\UL\\	
			43	.BYTE	67	
			002E5	.BYTE	68	
			002E6	.BLKB	2	
66	5B	1B	002E8	P.AAX:	.BYTE	27, 91, 102
			002EB	.BLKB	1	
48	1B	002EC	P.AAY:	.BYTE	27, 72	
			002EE	.BLKB	2	
41	1B	002F0	P.AAZ:	.ASCII	<27>\\A\\	
41	1B	002F2		.ASCII	<27>\\A\\	
41	1B	002F4		.ASCII	<27>\\A\\	

41	1B	002F6	.ASCII	<27>1A\
41	1B	002F8	.ASCII	<27>1A\
41	1B	002FA	.ASCII	<27>1A\
41	1B	002FC	.ASCII	<27>1A\
41	1B	002FE	.ASCII	<27>1A\
41	1B	00300	.ASCII	<27>1A\
41	1B	00302	.ASCII	<27>1A\
41	1B	00304	.ASCII	<27>1A\
41	1B	00306	.ASCII	<27>1A\
41	1B	00308	.ASCII	<27>1A\
41	1B	0030A	.ASCII	<27>1A\
41	1B	0030C	.ASCII	<27>1A\
41	1B	0030E	.ASCII	<27>1A\
41	1B	00310	.ASCII	<27>1A\
41	1B	00312	.ASCII	<27>1A\
41	1B	00314	.ASCII	<27>1A\
41	1B	00316	.ASCII	<27>1A\
41	1B	00318	.ASCII	<27>1A\
41	1B	0031A	.ASCII	<27>1A\
41	1B	0031C	.ASCII	<27>1A\
41	1B	0031E	.ASCII	<27>1A\
42	1B	00320	P.ABA:	.BYTE 27, 66
		00322		.BLKB 2
43	1B	00324	P.ABB:	.BYTE 27, 67

UP=	65
DOWN=	66
RIGHT=	67
UP_24=	P.AAZ

			OFFC 00000	.ENTRY		
					COB\$SET_CURSOR_REL, Save R2,R3,R4,R5,R6,-	2552
					R7,R8,R9,R10,R11	
					SYSSFAOL R11	
					P.AAU, R10	
					#56, SP	
					#17694726, UP CTL	
					P.AAU, UP CTL#4	
					#17694726, DOWN CTL	
					P.AAV, DOWN CTL#4	
					#17694726, RIGHT CTL	
					P.AAU, RIGHT CTL#4	
					TERM TYPE, R9	
					R9, #3	
					1S	
					R9, #2	
					1S	
					23S	
					CUR_SIZE, R6	
					(R6), BUFFER, FREE_ADDR	
					LINE_NO, R7	
					R8	
					R7	
					2S	
					R8	
					COL_NO	
					2S	

52	0C	AC	14	AC	C1 00061	ADDL3	COL_PLUS, COL_NO, R2	2744
54				56	D0 00067	MOVL	R6, R4	2743
53			18	AC	D0 0006A	MOVL	BUFFER, R3	
51				57	D0 0006E	MOVL	R7, R1	
50				59	D0 00071	MOVL	R9, R0	
				FEB1	30 00074	BSBW	COB\$SET_CURSOR_ABS R4	
55	18	AC		66	C1 00077	ADDL3	(R6), BUFFER, FREE_ADDR	
		03		58	E8 0007C	BLBS	R8, 48	
				00B9	31 0007F	BRW	148	2746
				0C	AC D5 00082	28:	TSTL	2749
				F8	12 00085	BNEQ	COL_NO	
				14	AC D5 00087	TSTL	38:	2750
				20	12 0008A	BNEQ	COL_PLUS	
				58	D4 0008C	CLRL	68	2753
		03		59	D1 0008E	CMPL	R8	
				10	12 00091	BNEQ	R9, #3	2756
				58	D6 00093	INCL	58	
85	18	00	18	AA	F0 00095	INSV	R8	
		55		02	C0 0009B	ADDL2	P_AAX, #0, #24, (FREE_ADDR)+	2757
		66		03	C0 0009E	ADDL2	#2, FREE_ADDR	
				4D	11 000A1	BRB	#3, (R6)	
		85	1C	AA	B0 000A3	MOVW	88	2756
		66		02	C0 000A7	ADDL2	P_AAY, (FREE_ADDR)+	2759
				44	11 000AA	BRB	#2, (R6)	
		03		58	D4 000AC	68:	88	2753
				59	D1 000AE	CLRL	R8	2769
				32	12 000B1	BNEQ	R9, #3	
				58	D6 000B3	INCL	78	
		18	AE	010E000F	18	DO 000B5	R8	
	1C	AE		8F	DO 000B8	MOVL	#24, CVT_ARG	2771
				55	DO 000C0	MOVL	#17694735, FAO_BUF	
				5E	DD 000C4	MOVL	FREE_ADDR, FAO_BUF+4	
				55	DD 000C6	PUSHL	SP	
				1C	AE 9F 000C6	PUSHAB	FAO_BUF	
				0C	AE 9F 000C9	PUSHAB	FAO_LEN	
				3C	AE 9F 000CC	PUSHAB	UP_CTL	
		6B		04	FB 000CF	CALLS	#4, SYS\$FAOL	
		43		50	E9 000D2	BLBC	STATUS, 98	
		50	04	AE	3C 000D5	MOVZWL	FAO_LEN, R0	
		66		50	C0 000D9	ADDL2	R0, (R6)	
		50	04	AE	3C 000DC	MOVZWL	FAO_LEN, R0	
		55		50	C0 000E0	ADDL2	R0, FREE_ADDR	
				0B	11 000E3	BRB	88	2769
65	20	AA		30	28 000E5	MOV3	#48, UP 24, (FREE_ADDR)	2774
		55		53	DO 000EA	MOVL	R3, FREE_ADDR	
		66		30	C0 000ED	ADDL2	#48, (R6)	2775
		39		58	E9 000F0	BLBC	R8, 118	2781
		01		57	D1 000F3	CMPL	R7, #1	2783
				43	13 000F6	BEQL	148	
		08	AE	A7	9E 000F8	MOVAB	-1(R7), CVT_ARG	
	18	AE	010E000F	8F	DO 000FD	MOVL	#17694735, FAO_BUF	
	1C	AE		55	DO 00105	MOVL	FREE_ADDR, FAO_BUF+4	
				08	AE 9F 00109	PUSHL	CVT_ARG	
				1C	AE 9F 0010C	PUSHL	FAO_BUF	
				14	AE 9F 0010F	PUSHL	FAO_LEN	
				34	AE 9F 00112	PUSHL	DOWN_CTL	
		6B		04	FB 00115	CALLS	#4, SYS\$FAOL	
	01			50	E8 00118	BLBS	STATUS, 108	



COB\$ESCAPE\_GEN COB\$ESCAPE\_GENERATOR - Escape sequence generat J 13  
1-003 COB\$SET\_CURSOR\_REL Create relative cursor posi 16-Sep-1984 00:06:34 VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBESCGEN.B32;1

Page 40  
(14)

F5 66 02 C0 001C1 ADDL2 #2, (R6)  
51 50 F3 001C4 228: AOBLEQ R0, COUNTER, 218  
50 01 D0 001C8 238: MOVL #1, R0  
04 001CB 248: RET

: 2821  
: 2823

; Routine Size: 460 bytes, Routine Base: \_COB\$CODE + 0326

; 1247 2824 1 !<BLF/PAGE>

```

1249      2825 1 %SBTTL 'COB$SETUP_TERM_TYPE - Setup terminal type for COB$ routines'
1250      2826 1 GLOBAL ROUTINE COB$SETUP_TERM_TYPE (
1251      2827 1   FILE_NAME,
1252      2828 1   NAME_LEN,
1253      2829 1   TERM_TYPE,
1254      2830 1   SEC_DEV_CHAR,
1255      2831 1   DEVICE_TYPE : REF VECTOR [,BYTE],
1256      2832 1   RES_NAME_LEN : REF VECTOR [,WORD],
1257      2833 1   RES_NAME_ADDR
1258      2834 1   )
1259      2835 1   ++
1260      2836 1   FUNCTIONAL DESCRIPTION:
1261      2837 1
1262      2838 1   This routine uses the specified file name to determine device
1263      2839 1   characteristics and assign a terminal type code which is understood
1264      2840 1   by other COB$ routines. COB$ routines use the terminal type to
1265      2841 1   determine the correct escape sequence for a given function (ex. set
1266      2842 1   cursor).
1267      2843 1
1268      2844 1   CALLING SEQUENCE:
1269      2845 1
1270      2846 1   ret_status.wlc.v = COB$SETUP_TERM_TYPE (FILE_NAME.rt.r,
1271      2847 1           NAME_LEN.rl.v,
1272      2848 1           TERM_TYPE.wl.r
1273      2849 1           [,SEC_DEV_CHAR.wlu.r]
1274      2850 1           [,DEVICE_TYPE.wbu.r]
1275      2851 1           [,RES_NAME_LEN.wwu.r]
1276      2852 1           RES_NAME_ADDR.wt.r)
1277      2853 1
1278      2854 1   FORMAL PARAMETERS:
1279      2855 1
1280      2856 1   FILE_NAME.rt.r      addr of file name text
1281      2857 1   NAME_LEN.rl.v      length of file name text
1282      2858 1   TERM_TYPE.wl.r      terminal type code, one of the following:
1283      2859 1
1284      2860 1           unknown
1285      2861 1           vt05
1286      2862 1           vt52
1287      2863 1           vt100
1288      2864 1           vtforeign
1289      2865 1           hardcopy
1290      2866 1   SEC_DEV_CHAR.wlu.r  [Optional] If supplied, the address of
1291      2867 1           a longword to receive the secondary
1292      2868 1           device dependent bits. This is the
1293      2869 1           field that, e.g. tells whether a VT100
1294      2870 1           has AVO.
1295      2871 1
1296      2872 1   DEVICE_TYPE.wbu.r  [Optional]. If present, address of byte
1297      2873 1           to receive hardware device type. These
1298      2874 1           are the DTS_type codes.
1299      2875 1
1300      2876 1   RES_NAME_LEN.wwu.r  [Optional -- if provided, RES_NAME_ADDR
1301      2877 1           must be provided as well.] If present,
1302      2878 1           the address of a word to receive the
1303      2879 1           length of the resultant name string.
1304      2880 1
1305      2881 1   RES_NAME_ADDR.wt.r  [Optional -- if provided, RES_NAME_LEN

```

L 13

1306 2882 1 | must be provided as well.] If present,  
 1307 2883 1 | the address of a buffer to receive the  
 1308 2884 1 | resultant name string. NOTE: This  
 1309 2885 1 | routine assumes that the supplied buffer  
 1310 2886 1 | is large enough to contain the resultant  
 1311 2887 1 | name string. It must be a minimum of 4  
 1312 2888 1 | bytes long and should be at least 64  
 1313 2889 1 | bytes long to guarantee that the name  
 1314 2890 1 | will fit.

1315 2891 1 | IMPLICIT INPUTS:  
 1316 2892 1 | NONE

1317 2893 1 | IMPLICIT OUTPUTS:  
 1318 2894 1 | NONE

1319 2895 1 | COMPLETION STATUS:  
 1320 2896 1 |  
 1321 2897 1 | SIDE EFFECTS:  
 1322 2898 1 | NONE  
 1323 2899 1 |  
 1324 2900 1 |  
 1325 2901 1 |  
 1326 2902 1 |  
 1327 2903 1 |  
 1328 2904 1 |  
 1329 2905 1 |  
 1330 2906 1 |  
 1331 2907 2 | BEGIN  
 1332 2908 2 |  
 1333 2909 2 | BUILTIN  
 1334 2910 2 | NULLPARAMETER;  
 1335 2911 2 |  
 1336 2912 2 | LOCAL  
 1337 2913 2 | DEVNAM DSC : BLOCK [8, BYTE],  
 1338 2914 2 | DVI\_ITMLST : VECTOR [3\*3 + 1] INITIAL, | dsc for name  
 1339 2915 2 | (DVIS\_DEVTYPE \* 16 + 4, 0, 0, | item list for \$GETDVI  
 1340 2916 2 | DVIS\_DEVDEPEND2 \* 16 + 4, 0, 0, | device type  
 1341 2917 2 | DVIS\_DEVNAM \* 16 + 64, 0, 0, | device dependent bits  
 1342 2918 2 | 0), | result name string  
 1343 2919 2 | | terminator  
 1344 2920 2 | DVI\_EFN, | event flag for \$GETDVI.  
 1345 2921 2 | STATUS, | status retd by called routines  
 1346 2922 2 | DEV\_TYPE : VOLATILE, | storage for \$GETDVI value  
 1347 2923 2 | DEV\_DEPEND2 : VOLATILE, | storage for \$GETDVI value  
 1348 2924 2 |  
 1349 2925 2 | DEV\_DEVNAM : VECTOR [64, BYTE], | Buffer for result name  
 1350 2926 2 |  
 1351 2927 2 |  
 1352 2928 2 | DEV\_NAMLEN : VOLATILE WORD; | string  
 1353 2929 2 |  
 1354 2930 2 | BIND | Length of returned  
 1355 2931 2 | DVI\_TYPE = DVI\_ITMLST + 4, | resultant name string  
 1356 2932 2 | DVI\_DEPEND2 = DVI\_ITMLST + 16, | make it easy to reference  
 1357 2933 2 | DVI\_DEVNAM = DVI\_ITMLST + 28, | items retd by \$GETDVI  
 1358 2934 2 | DVI\_NAMLEN = DVI\_ITMLST + 32; |  
 1359 2935 2 |  
 1360 2936 2 | MAP |  
 1361 2937 2 | DEV\_DEPEND2 : BLOCK [4, BYTE]; |  
 1362 2938 2 |

```

1363 2939 2 DVI_TYPE = DEV_TYPE;           ! fill in rest of itmlst
1364 2940 2 DVI_DEPEND2 = DEV_DEPEND2;
1365 2941 2 DVI_DEVNAM = DEV_DEVNAM;
1366 2942 2 DVI_NAMLEN = DEV_NAMLEN;
1367 2943 2
1368 2944 2 IF NOT (STATUS = LIBSGET_EF (DVI_EFN))
1369 2945 2 THEN RETURN (.STATUS);          ! get unique event flag number
1370 2946 2
1371 2947 2 DEVNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1372 2948 2 DEVNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
1373 2949 2 DEVNAM_DSC [DSC$U_LENGTH] = .NAME_LEN;
1374 2950 2 DEVNAM_DSC [DSC$A_POINTER] = .FILE_NAME; ! dsc needed for SGETDVI
1375 2951 2
1376 P 2952 2 STATUS = SGETDVI (EFN = .DVI_EFN, DEVNAM = DEVNAM_DSC,
1377 2953 2           ITMLST = DVI_ITMLST);
1378 2954 2 IF NOT .STATUS THEN RETURN (.STATUS);
1379 2955 2
1380 2956 2 SWAITFR (EFN = .DVI_EFN);        ! make SGETDVI synchronous
1381 2957 2
1382 2958 2 IF NOT (STATUS = LIBSFREE_EF (DVI_EFN))
1383 2959 2 THEN RETURN (.STATUS);          ! free event flag
1384 2960 2
1385 2961 2 SELECTONE .DEV_TYPE OF
1386 2962 2 SET
1387 2963 2   [DTS_VT100]:
1388 2964 2     .TERM_TYPE = VT100;
1389 2965 2
1390 2966 2   [DTS_VT52, DTS_VT55]:
1391 2967 2     .TERM_TYPE = VT52;
1392 2968 2
1393 2969 2   [DTS_VT05]:
1394 2970 2     .TERM_TYPE = VT05;
1395 2971 2
1396 2972 2   [DTS_FT1 TO DTS_FT2]:
1397 2973 2     .TERM_TYPE = VTFOREIGN;
1398 2974 2
1399 2975 2   [DTS_LA36, DTS_LA120, DTS_LA34, DTS_LA38]:
1400 2976 2     .TERM_TYPE = HARDCOPY;
1401 2977 2
1402 2978 2   [OTHERWISE]:
1403 2979 2     IF .DEV_DEPEND2 [TT2$V_DECCRT] OR
1404 2980 2       .DEV_DEPEND2 [TT2$V_ANSICRT]
1405 2981 2     THEN
1406 2982 2       .TERM_TYPE = VT100      ! VT100 compatible (ANSI)
1407 2983 2     ELSE
1408 2984 2       .TERM_TYPE = UNKNOWN;  ! really unknown
1409 2985 2
1410 2986 2   TES:
1411 2987 2
1412 2988 2   !+ Return optional parameters if requested.
1413 2989 2
1414 2990 2   IF NOT NULLPARAMETER (4)
1415 2991 2     THEN
1416 2992 2       .SEC_DEV_CHAR = .DEV_DEPEND2;
1417 2993 2
1418 2994 2   IF NOT NULLPARAMETER (5)
1419 2995 2     THEN

```

```
1420      2996 2
1421      2997 2
1422      2998 2
1423      2999 2
1424      3000 2
1425      3001 2
1426      3002 2
1427      3003 2
1428      3004 2
1429      3005 2
1430      3006 2
1431      3007 2
                DEVICE_TYPE [0] = .DEV_TYPE;
                IF NOT NULLPARAMETER (6) AND
                   NOT NULLPARAMETER (7)
                THEN
                  BEGIN
                    CHSMOVE (.DEV_NAMLEN, DEV_DEVNAM, .RES_NAME_ADDR);
                    RES_NAME_LEN [0] = .DEV_NAMLEN;
                  END;
                RETURN (.STATUS);
                END;
                ! End of routine C0
```

! End of routine COBSSSETUP\_TERM\_TYPE

0C	BC	06	14	0008F	BGTR	3\$
		02	D0	00091	MOVL	#2, @TERM_TYPE
	01	3D	11	00095	BRB	9\$
		50	D1	00097	CMPL	R0, #1
0C	BC	06	12	0009A	BNEQ	4\$
		01	D0	0009C	MOVL	#1, @TERM_TYPE
	10	32	11	000A0	BRB	9\$
		50	D1	000A2	CMPL	R0, #16
	11	08	19	000A5	BLSS	5\$
		50	D1	000A7	CMPL	R0, #17
0C	BC	06	14	000AA	BGTR	5\$
		04	D0	000AC	MOVL	#4, @TERM_TYPE
	20	22	11	000B0	BRB	9\$
		50	D1	000B2	CMPL	R0, #32
	23	08	19	000B5	BLSS	6\$
		50	D1	000B7	CMPL	R0, #35
0C	BC	06	14	000BA	BGTR	6\$
		05	D0	000BC	MOVL	#5, @TERM_TYPE
		12	11	000C0	BRB	9\$
04	4B	AE	05	E0 000C2	6\$:	BBS
	06	4B	AE	E9 000C7		BLBC
0C	BC	03	D0	000CB	7\$:	MOVL
		03	11	000CF		BRB
	04	0C	BC	D4 000D1	8\$:	CLRL
		6C	91	000D4	9\$:	CMPB
		0A	1F	000D7		BLSSU
		10	AC	D5 000D9		TSTL
		05	13	000DC		BEQL
10	BC	48	AE	D0 000DE		MOVL
	05	6C	91	000E3	10\$:	CMPB
		0A	1F	000E6		BLSSU
		14	AC	D5 000E8		TSTL
		05	13	000EB		BEQL
14	BC	4C	AE	90 000ED		MOVB
	06	6C	91	000F2	11\$:	CMPB
		1B	1F	000F5		BLSSU
		18	AC	D5 000F7		TSTL
		16	13	000FA		BEQL
	07	6C	91	000FC		CMPB
		11	1F	000FF		(AP), #7
		1C	AC	D5 00101		BLSSU
		0C	13	00104		TSTL
1C	BC	08	AE	28 00106		BEQL
	18	BC	06	AE B0 0010D		MOVC3
		50	D0	00112	12\$:	MOVW
		04	D0	00115		MOVL
						RFT

; Routine Size: 278 bytes, Routine Base: \_COB\$CODE + 051C

1432 3008 1 !<BLF/PAGE>

```
1434      3009 1 %SBTTL 'COB$SUP_SCROLL_R2 - Create up scroll sequence'  
1435      3010 1 GLOBAL ROUTINE COB$SUP_SCROLL_R2 (   
1436          TERM_TYPE,  
1437          BUFFER,  
1438          CUR_SIZE  
1439          ) : COB$SESC_R2_LNK =  
1440      3015 1 ++  
1441      3016 1 FUNCTIONAL DESCRIPTION:  
1442      3017 1  
1443      3018 1 This routine generates the escape sequence for up scroll.  
1444      3019 1 The string is appended into the buffer.  
1445      3020 1  
1446      3021 1 CALLING SEQUENCE:  
1447      3022 1  
1448      3023 1     ret_status.wlc.v = COB$SUP_SCROLL_R2 (TERM_TYPE.rl.v, BUFFER.mt.r,  
1449      3024 1                 CUR_SIZE.ml.r)  
1450      3025 1  
1451      3026 1 FORMAL PARAMETERS:  
1452      3027 1  
1453      3028 1     TERM_TYPE.rl.v          terminal type  
1454      3029 1     BUFFER.mt.r          addr of buffer  
1455      3030 1     CUR_SIZE.ml.r          # bytes currently in buffer  
1456      3031 1  
1457      3032 1 IMPLICIT INPUTS:  
1458      3033 1     NONE  
1459      3034 1  
1460      3035 1  
1461      3036 1 IMPLICIT OUTPUTS:  
1462      3037 1     NONE  
1463      3038 1  
1464      3039 1  
1465      3040 1 COMPLETION STATUS:  
1466      3041 1  
1467      3042 1  
1468      3043 1  
1469      3044 1  
1470      3045 1  
1471      3046 1  
1472      3047 1  
1473      3048 2  
1474      3049 2  
1475      3050 2  
1476      3051 2 LOCAL  
1477      3052 2     FREE_ADDR : REF VECTOR [,BYTE];  
1478      3053 2  
1479      3054 2  
1480      3055 2  
1481      3056 2  
1482      3057 2 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF  
1483      3058 3  
1484      3059 3  
1485      3060 3  
1486      3061 3  
1487      3062 3  
1488      3063 3  
1489      3064 3  
1490      3065 2  
1491      3066 2  
1492      3067 2  
1493      3068 2  
1494      3069 2  
1495      3070 2  
1496      3071 2  
1497      3072 2  
1498      3073 2  
1499      3074 2  
1500      3075 2  
1501      3076 2  
1502      3077 2  
1503      3078 2  
1504      3079 2  
1505      3080 2  
1506      3081 2  
1507      3082 2  
1508      3083 2  
1509      3084 2  
1510      3085 2  
1511      3086 2  
1512      3087 2  
1513      3088 2  
1514      3089 2  
1515      3090 2  
1516      3091 2  
1517      3092 2  
1518      3093 2  
1519      3094 2  
1520      3095 2  
1521      3096 2  
1522      3097 2  
1523      3098 2  
1524      3099 2  
1525      3100 2  
1526      3101 2  
1527      3102 2  
1528      3103 2  
1529      3104 2  
1530      3105 2  
1531      3106 2  
1532      3107 2  
1533      3108 2  
1534      3109 2  
1535      3110 2  
1536      3111 2  
1537      3112 2  
1538      3113 2  
1539      3114 2  
1540      3115 2  
1541      3116 2  
1542      3117 2  
1543      3118 2  
1544      3119 2  
1545      3120 2  
1546      3121 2  
1547      3122 2  
1548      3123 2  
1549      3124 2  
1550      3125 2  
1551      3126 2  
1552      3127 2  
1553      3128 2  
1554      3129 2  
1555      3130 2  
1556      3131 2  
1557      3132 2  
1558      3133 2  
1559      3134 2  
1560      3135 2  
1561      3136 2  
1562      3137 2  
1563      3138 2  
1564      3139 2  
1565      3140 2  
1566      3141 2  
1567      3142 2  
1568      3143 2  
1569      3144 2  
1570      3145 2  
1571      3146 2  
1572      3147 2  
1573      3148 2  
1574      3149 2  
1575      3150 2  
1576      3151 2  
1577      3152 2  
1578      3153 2  
1579      3154 2  
1580      3155 2  
1581      3156 2  
1582      3157 2  
1583      3158 2  
1584      3159 2  
1585      3160 2  
1586      3161 2  
1587      3162 2  
1588      3163 2  
1589      3164 2  
1590      3165 2  
1591      3166 2  
1592      3167 2  
1593      3168 2  
1594      3169 2  
1595      3170 2  
1596      3171 2  
1597      3172 2  
1598      3173 2  
1599      3174 2  
1600      3175 2  
1601      3176 2  
1602      3177 2  
1603      3178 2  
1604      3179 2  
1605      3180 2  
1606      3181 2  
1607      3182 2  
1608      3183 2  
1609      3184 2  
1610      3185 2  
1611      3186 2  
1612      3187 2  
1613      3188 2  
1614      3189 2  
1615      3190 2  
1616      3191 2  
1617      3192 2  
1618      3193 2  
1619      3194 2  
1620      3195 2  
1621      3196 2  
1622      3197 2  
1623      3198 2  
1624      3199 2  
1625      3200 2  
1626      3201 2  
1627      3202 2  
1628      3203 2  
1629      3204 2  
1630      3205 2  
1631      3206 2  
1632      3207 2  
1633      3208 2  
1634      3209 2  
1635      3210 2  
1636      3211 2  
1637      3212 2  
1638      3213 2  
1639      3214 2  
1640      3215 2  
1641      3216 2  
1642      3217 2  
1643      3218 2  
1644      3219 2  
1645      3220 2  
1646      3221 2  
1647      3222 2  
1648      3223 2  
1649      3224 2  
1650      3225 2  
1651      3226 2  
1652      3227 2  
1653      3228 2  
1654      3229 2  
1655      3230 2  
1656      3231 2  
1657      3232 2  
1658      3233 2  
1659      3234 2  
1660      3235 2  
1661      3236 2  
1662      3237 2  
1663      3238 2  
1664      3239 2  
1665      3240 2  
1666      3241 2  
1667      3242 2  
1668      3243 2  
1669      3244 2  
1670      3245 2  
1671      3246 2  
1672      3247 2  
1673      3248 2  
1674      3249 2  
1675      3250 2  
1676      3251 2  
1677      3252 2  
1678      3253 2  
1679      3254 2  
1680      3255 2  
1681      3256 2  
1682      3257 2  
1683      3258 2  
1684      3259 2  
1685      3260 2  
1686      3261 2  
1687      3262 2  
1688      3263 2  
1689      3264 2  
1690      3265 2  
1691      3266 2  
1692      3267 2  
1693      3268 2  
1694      3269 2  
1695      3270 2  
1696      3271 2  
1697      3272 2  
1698      3273 2  
1699      3274 2  
1700      3275 2  
1701      3276 2  
1702      3277 2  
1703      3278 2  
1704      3279 2  
1705      3280 2  
1706      3281 2  
1707      3282 2  
1708      3283 2  
1709      3284 2  
1710      3285 2  
1711      3286 2  
1712      3287 2  
1713      3288 2  
1714      3289 2  
1715      3290 2  
1716      3291 2  
1717      3292 2  
1718      3293 2  
1719      3294 2  
1720      3295 2  
1721      3296 2  
1722      3297 2  
1723      3298 2  
1724      3299 2  
1725      3300 2  
1726      3301 2  
1727      3302 2  
1728      3303 2  
1729      3304 2  
1730      3305 2  
1731      3306 2  
1732      3307 2  
1733      3308 2  
1734      3309 2  
1735      3310 2  
1736      3311 2  
1737      3312 2  
1738      3313 2  
1739      3314 2  
1740      3315 2  
1741      3316 2  
1742      3317 2  
1743      3318 2  
1744      3319 2  
1745      3320 2  
1746      3321 2  
1747      3322 2  
1748      3323 2  
1749      3324 2  
1750      3325 2  
1751      3326 2  
1752      3327 2  
1753      3328 2  
1754      3329 2  
1755      3330 2  
1756      3331 2  
1757      3332 2  
1758      3333 2  
1759      3334 2  
1760      3335 2  
1761      3336 2  
1762      3337 2  
1763      3338 2  
1764      3339 2  
1765      3340 2  
1766      3341 2  
1767      3342 2  
1768      3343 2  
1769      3344 2  
1770      3345 2  
1771      3346 2  
1772      3347 2  
1773      3348 2  
1774      3349 2  
1775      3350 2  
1776      3351 2  
1777      3352 2  
1778      3353 2  
1779      3354 2  
1780      3355 2  
1781      3356 2  
1782      3357 2  
1783      3358 2  
1784      3359 2  
1785      3360 2  
1786      3361 2  
1787      3362 2  
1788      3363 2  
1789      3364 2  
1790      3365 2  
1791      3366 2  
1792      3367 2  
1793      3368 2  
1794      3369 2  
1795      3370 2  
1796      3371 2  
1797      3372 2  
1798      3373 2  
1799      3374 2  
1800      3375 2  
1801      3376 2  
1802      3377 2  
1803      3378 2  
1804      3379 2  
1805      3380 2  
1806      3381 2  
1807      3382 2  
1808      3383 2  
1809      3384 2  
1810      3385 2  
1811      3386 2  
1812      3387 2  
1813      3388 2  
1814      3389 2  
1815      3390 2  
1816      3391 2  
1817      3392 2  
1818      3393 2  
1819      3394 2  
1820      3395 2  
1821      3396 2  
1822      3397 2  
1823      3398 2  
1824      3399 2  
1825      3400 2  
1826      3401 2  
1827      3402 2  
1828      3403 2  
1829      3404 2  
1830      3405 2  
1831      3406 2  
1832      3407 2  
1833      3408 2  
1834      3409 2  
1835      3410 2  
1836      3411 2  
1837      3412 2  
1838      3413 2  
1839      3414 2  
1840      3415 2  
1841      3416 2  
1842      3417 2  
1843      3418 2  
1844      3419 2  
1845      3420 2  
1846      3421 2  
1847      3422 2  
1848      3423 2  
1849      3424 2  
1850      3425 2  
1851      3426 2  
1852      3427 2  
1853      3428 2  
1854      3429 2  
1855      3430 2  
1856      3431 2  
1857      3432 2  
1858      3433 2  
1859      3434 2  
1860      3435 2  
1861      3436 2  
1862      3437 2  
1863      3438 2  
1864      3439 2  
1865      3440 2  
1866      3441 2  
1867      3442 2  
1868      3443 2  
1869      3444 2  
1870      3445 2  
1871      3446 2  
1872      3447 2  
1873      3448 2  
1874      3449 2  
1875      3450 2  
1876      3451 2  
1877      3452 2  
1878      3453 2  
1879      3454 2  
1880      3455 2  
1881      3456 2  
1882      3457 2  
1883      3458 2  
1884      3459 2  
1885      3460 2  
1886      3461 2  
1887      3462 2  
1888      3463 2  
1889      3464 2  
1890      3465 2  
1891      3466 2  
1892      3467 2  
1893      3468 2  
1894      3469 2  
1895      3470 2  
1896      3471 2  
1897      3472 2  
1898      3473 2  
1899      3474 2  
1900      3475 2  
1901      3476 2  
1902      3477 2  
1903      3478 2  
1904      3479 2  
1905      3480 2  
1906      3481 2  
1907      3482 2  
1908      3483 2  
1909      3484 2  
1910      3485 2  
1911      3486 2  
1912      3487 2  
1913      3488 2  
1914      3489 2  
1915      3490 2  
1916      3491 2  
1917      3492 2  
1918      3493 2  
1919      3494 2  
1920      3495 2  
1921      3496 2  
1922      3497 2  
1923      3498 2  
1924      3499 2  
1925      3500 2  
1926      3501 2  
1927      3502 2  
1928      3503 2  
1929      3504 2  
1930      3505 2  
1931      3506 2  
1932      3507 2  
1933      3508 2  
1934      3509 2  
1935      3510 2  
1936      3511 2  
1937      3512 2  
1938      3513 2  
1939      3514 2  
1940      3515 2  
1941      3516 2  
1942      3517 2  
1943      3518 2  
1944      3519 2  
1945      3520 2  
1946      3521 2  
1947      3522 2  
1948      3523 2  
1949      3524 2  
1950      3525 2  
1951      3526 2  
1952      3527 2  
1953      3528 2  
1954      3529 2  
1955      3530 2  

```

```
1491      3066 2 [VT52, VT100]:  
1492      3067 2 BEGIN  
1493      3068 3 FREE_ADDR [0] = LF;  
1494      3069 3 .CUR_SIZE = ..CUR_SIZE + 1;  
1495      3070 2 END;  
1496      3071 2  
1497      3072 2 [HARDCOPY, UNKNOWN, VTFOREIGN]:  
1498      3073 2 :  
1499      3074 2  
1500      3075 2 [INRANGE, OUTRANGE]:  
1501      3076 2 RETURN 0;          ! should never get here  
1502      3077 2  
1503      3078 2 TES;  
1504      3079 2  
1505      3080 2 RETURN (SSS_NORMAL);  
1506      3081 2  
1507      3082 1 END;          ! End of routine COBSSUP_SCROLL_R
```

		51	62	CO 00000 COB\$SUP_SCROLL_R2::	
0016	05	00	50	CF 00003	ADDL2 (CUR_SIZE), FREE_ADDR
	0016	000E	001B	00007	CASEL TERM_TYPE, NO. #5
		001B	0000F	1\$: .WORD	4S-1\$,-
					2S-1\$,-
					3S-1\$,-
					3S-1\$,-
					4S-1\$,-
					4S-1\$,-
			11	11 00013	BRB 5\$
		61	0A	D0 00015	MOVL #10, (FREE_ADDR)
		62	04	C0 00018	ADDL2 #4, (CUR_SIZE)
			05	11 0001B	BRB 4S
		61	0A	90 0001D	MOVB #10, (FREE_ADDR)
			62	D6 00020	INCL (CUR_SIZE)
		50	01	D0 00022	MOVL #1, R0
			05	00025	RSB
			50	D4 00026	CLRL R0
			05	00028	RSB

; Routine Size: 41 bytes, Routine Base: COB\$CODE + 0632

3083 1 !<BLF/PAGE>

COB\$ESCAPE\_GEN COB\$ESCAPE GENERATOR - Escape sequence generat E 14  
 1-003 COB\$UP\_SCROLL\_R2 - Create up scroll sequence 16-Sep-1984 00:06:34 VAX-11 Bliss-32 V4.0-742  
 : 1510 3084 1 END [COBRTL.SRC]COBESCGEN.B32;1  
 : 1511 3085 1  
 : 1512 3086 0 ELUDOM ! End of module COB\$ESCAPE\_GENERATOR

Page 48  
(17)

#### PSECT SUMMARY

Name	Bytes	Attributes
_COB\$CODE	1627	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

#### Library Statistics

File	-----	Symbols	-----	Pages	Processing
	Total	Loaded	Percent	Mapped	Time
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	30	0	581	00:00.7
-\$255\$DUA28:[COBRTL.OBJ]SMGLIB.L32;1	469	31	6	38	00:00.2

#### COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:COBESCGEN/OBJ=OBJ\$:COBESCGEN MSRC\$:COBESCGEN/UPDATE=(ENH\$:COBESCGEN )

: Size: 1396 code + 231 data bytes  
 : Run Time: 00:24.7  
 : Elapsed Time: 01:33.0  
 : Lines/CPU Min: 7484  
 : Lexemes/CPU-Min: 27092  
 : Memory Used: 234 pages  
 : Compilation Complete

0062 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

